VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Nástroj pro překlad výrazů relační algebry do SQL

Tool for Translation of Relation Algebra Expressions into SQL

2012 Pavel Brziák

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Pavel Brziák**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Nástroj pro překlad výrazů relační algebry do SQL
Tool for Translation of Relational Algebra Expressions into SQL

Zásady pro vypracování:

Student má za úkol navrhnout a implementovat aplikaci "tvůrce výrazů v relační algebře" s uživatelsky přívětivým prostředím pro vytvoření, případně načtení výrazu v relační algebře nad zvolenými tabulkami vybraného SŘBD. K tomuto výrazu bude automaticky generován odpovídající příkaz SELECT v jazyce SQL a zobrazen výsledek SQL příkazu. Aplikace dále umožní načtení nebo vytvoření jednoduché databáze ve zvoleném SŘBD a vkládání a rušení dat pro testovací účely.
Student vypracuje analýzu, návrh implementace a implementaci ve zvoleném programovacím jazyce s podporou vhodného SŘBD.

1. Prostudujte příkazy relační algebry a jazyka SQL a specifikujte požadavky na program.
2. Navrhněte uživatelské prostředí pro "tvůrce výrazů v relační algebře" - načtení relačního schématu, zadávání dotazů nad relacemi v relační algebře, vygenerování odpovídajícího příkazu SELECT v SQL jazyce a zobrazení výsledné množiny dat.
3. Implementujte navržené uživatelské prostředí ve zvoleném programovacím jazyce s podporou vhodného SŘBD.
4. Program otestujte na vhodně zvolených příkladech, napište programátorskou a uživatelskou příručku.
5. Porovnejte s existujícími aplikacemi.

Seznam doporučené odborné literatury:

[1] POKORNÝ, Jaroslav. *Dotazovací jazyky* Vydání první. Veletiny: SCIENCE, 1994. 228 s. ISBN 80-901475-2-6
[2] ŠARMANOVÁ, Jana. *Teorie zpracování dat* [online]. druhé, přepracované. Ostrava : VŠB – Technická univerzita Ostrava, 2007 [cit. 2010-11-09]. Dostupné z WWW: <http://www.elearn.vsb.cz/archivcd/FEI/TZD/TZD.pdf>. ISBN 978-80-248-1498-8

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Emilie Šeptáková**

Datum zadání: 18.11.2011

Datum odevzdání: 04.05.2012

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 4. května 2012 .....

Poděkování

Rád bych poděkoval vedoucí mé bakalářské práce, paní Ing. Emilii Šeptákové, za pomoc při vytváření této bakalářské práce a její hodnotné připomínky a příspěvky.

Abstrakt

Cílem této práce je implementace nástroje pro překlad výrazů relační algebry. Tento nástroj má dynamicky převádět algoritmus v relační algebře na shodný dotaz v jazyce SQL. Čtenář se seznámí se syntaxí relační algebry, jejími pravidly a funkcí. Je mu krátce představen jazyk SQL a možnosti jeho použití. Součástí práce je popis překladu výrazů relační algebry a dále popis návrhu implementace programu nad zvoleným databázovým systémem. Práce se dále zabývá funkcemi implementovaného programu a popisem jejich realizace. V závěru práce je program testován a výsledky jsou prezentovány, implementovaná aplikace je porovnána s podobnými již existujícími programy.

Abstract

Aim of this thesis is implementation of tool for translation expressions in relational algebra. This tool should dynamically convert algorithm in relational algebra into equal SQL query. The reader will learn the syntax of relational algebra, its rules and function. He's being briefly introduced to SQL and possibilities of its use. Part of this work is description of translation relational algebra expressions and description of implementation design on selected database system. Thesis further describes functions of implemented application and contains description of its realization. The last part is about the application testing and the results are presented. Implemented application is compared with similar already existing applications.

Klíčová slova

Relační algebra, SQL, Překlad relační algebry, Oracle database, C#

Keywords

Relational algebra, SQL, Relational algebra translation, Oracle database, C#

Seznam použitých zkratek a symbolů

Admin	–	Uživatel s právy administrátora
ERD	–	Entity-relationship diagram
MSSQL	–	Microsoft SQL Server
MySQL	–	Databázový systém MySQL
NU	–	Nepřihlášený uživatel
Oracle	–	Oracle database
RMD	–	Relační model dat
SQL	–	Structured Query Language
SŘBD	–	Systém řízení báze dat

Seznam tabulek

Seznam obrázků

Obsah

1. Úvod

Jako téma své bakalářské práce jsem zvolil vytvoření nástroje pro překlad výrazů relační algebry do SQL. Toto téma jsem si vybral z důvodu, že mi teorie relační databází přijde velice zajímavá a setkal jsem se s ní již v mnoha předmětech, kde bylo toto téma probíráno včetně praktické implementace aplikací komunikujících s databázemi. Doufal jsem, že mi výběr tohoto tématu umožní hlubší nahlédnutí do problematiky relačních databází a relační algebry, která je mnohými považována za základ pro práci s databázovými systémy.

V této práci se nejdříve budu zabývat teorií překladu výrazů relační algebry do SQL. Dalším krokem bude výběr vhodného SŘBD (systém řízení báze dat), nad kterým budou přeložené příkazy v jazyce SQL prováděny. Dle možností vybraného SŘBD vytvořím program, jehož záměrem bude poskytnutí přívětivého rozhraní, pomocí kterého bude moci uživatel snadno vytvářet i složité výrazy relační algebry a porovnávat tyto výrazy s výsledným vygenerovaným dotazem v SQL. Tohoto by mělo být dosaženo pomocí intuitivního tvůrce výrazů. Samozřejmostí je také zobrazení výsledné množiny dat nad zadaným výrazem. Implementovaný program dále umožní dynamicky vytvářet a rušit tabulky podle zadaných kritérií a vkládat data. Další možností uživatele bude ukládání výrazů relační algebry a jejich opětovné načtení z databáze. Cílem je vytvořit intuitivní nástroj, který uživatelům umožní přeložit výrazy relační algebry, zobrazit tyto přeložené výrazy v jazyce SQL a výslednou relaci dle přeloženého dotazu.

Čtenář se při čtení této práce seznámí s relační algebrou, SQL a návrhem a implementací aplikace, kterou jsem vytvářel. Součástí této práce je také programátorská a uživatelská příručka, ve které se může čtenář dozvědět další podrobnosti o implementovaném programu.

2. Podrobná specifikace zadání

V této části specifikuji zadání svého programu, který se poté pokusím naimplementovat.

2.1. Cíle programu

Cílem implementovaného programu by mělo být umožnění přihlášení uživatele k databázi a nad tabulkami databáze, ke kterým má uživatel přístup umožnit jednoduše vytvářet výrazy v relační algebře v přehledném uživatelském rozhraní. Samozřejmostí by měla být kontrola vkládaných výrazů a případné upozornění uživatele na chyby v nich. Program by měl být schopen výrazy relační algebry korektně přeložit do jazyka SQL a zobrazit výslednou relaci podle výrazu relační algebry. Dále by měl program umožňovat ukládat výrazy relační algebry a jejich načítaní, mazání. Jedním z požadavků také je, aby program uměl vytvářet uživatelem definované tabulky v databázi, mazat je a plnit testovacími daty.

2.2. Role v systému

Program bude mít režimy dle práv uživatele. Tato práva bude třeba získat ze SŘBD. Systém by tedy měl umožnit přihlášení uživatele ke konkrétnímu SŘBD, poté poslat dotaz na jeho práva k databázi a dle obdržených informací zjistit, zda můžeme přihlášeného uživatele popsat jako administrátora, nebo jako běžného uživatele. Pokud uživatel nemá přístup k databázi, měl by program umožňovat přepnutí do offline překladače, který nijak nepracuje s databází.

1. Nepřihlášený uživatel – Má přístup k offline překladu výrazů relační algebry do SQL. Nepracuje nijak se SŘBD.
2. Uživatel – po přihlášení k databázi může vytvářet operace relační algebry nad tabulkami, dále může skládat tyto operace v samotné výrazy relační algebry. Po vytvoření operace nad tabulkou je tato dynamicky přeložena do SQL. Po poskládání konkrétního výrazu relační algebry má možnost ho uložit do databáze, zobrazit již uložené výrazy, mazat je a načítat je zpět z databáze. Vzorec může přeložit do jazyka SQL, výsledný dotaz a relace je mu zobrazena.
3. Administrátor – má všechny možnosti práce s programem jako uživatel. Je mu ale také zobrazena nabídka úpravy tabulek – může vytvářet nové tabulky, mazat je, vkládat a mazat záznamy a vymazat uložené dotazy relační algebry všech uživatelů.

2.3. Vstupy

Přihlášení uživatele do SŘBD
Program by měl poskytovat možnost přihlášení ke konkrétnímu SŘBD.
Možnost vytvoření operace nad tabulkou
Na program je kladen požadavek mít možnost si vybrat libovolnou z tabulek, ke které má uživatel přístup a nastavit nad ní požadovanou projekci a selekci.
Možnost vytvoření výrazu relační algebry
Program by měl umožňovat z jednotlivých operací a operandů relační algebry vytvořit kompletní výraz relační algebry ve tvůrci výrazů.
Možnost vytvoření výrazu relační algebry bez tvůrce výrazů
Program by měl umožňovat vkládání výrazů relační algebry ve formě textového řetězce.
Možnost uložení výrazu
Program by dále měl umožňovat vytvořený vzorec uložit se zadanými parametry, znovu jej načíst a případně otestovat.

2.4. Výstupy

Zobrazení všech tabulek a jejích sloupců, ke kterým má uživatel přístup.
Zobrazení výrazu v relační algebře.
Zobrazení přeloženého příkazu v SQL.
Zobrazení výsledné relace dle zadaného výrazu relační algebry.

2.5. Kontrola správnosti výrazů

Kontrola správnosti výrazů relační algebry by měla probíhat ve dvou krocích. V prvé řadě by měl program sám obsahovat kontrolu správnosti zadaných výrazů relační algebry. Uživatel nebude moct např. provést operace nad tabulkami, které neexistují, nebude moct použít sloupce, které se ve vybrané tabulce nevyskytují, nebude umožněno vytvářet podmínky, které jsou logicky nesprávné.

Pokud bude výraz formálně správně, ale SŘBD ho z nějakého důvodu vyhodnotí jako chybný, bude tato výjimka zachytávána v programu a zobrazena uživateli. Nejčastější výjimky se pokusím přeložit z Angličtiny a popsat důvod jejich vzniku uživateli.

2.6. Softwarové požadavky

Pro spuštění programu bude třeba splnit následující požadavky na hardware a software:

Hardware
Procesor min. 1,8 GHz
RAM min. 512 MB

Software
Windows XP – Windows 7

2.7. Pro uživatele

Pro uživatele bude k dispozici uživatelská příručka, ve které budou podrobně popsány možnosti programu včetně obrázků a příkladů práce. Dále se program pokusí co nejpřesněji zobrazovat důvody případných chyb.

2.8. Pro programátory

Pro programátory bude k dispozici programátorská příručka, ve které bude program popsán z programátorského hlediska.

2.9. Testování aplikace

O testování aplikace poprosím svou vedoucí bakalářské práce, která má velké zkušenosti s praktickou funkcí databázových systémů. V jedné z posledních částí bakalářské práce také vytvořím pomocí své aplikace testovací tabulky, nad kterými vytvořím netriviální výrazy relační algebry. Budu sledovat, jak úspěšně program tyto výrazy přeloží a výsledky vyhodnotím.

3. Relační algebra

Je třeba vzít v úvahu, že přestože je syntaxe relační algebry určena velice přesně, velké množství relevantních zdrojů si ji upravuje z důvodu, že složitější výrazy v relační algebře jsou pro nezkušené čtenáře nepřehledné. Při přehledu operací relační algebry se pokusím čtenáře seznámit s neupravovanou syntaxí. Protože je ale cílem této práce vytvoření nástroje pro překlad, považuji mnohem důležitější přesné pochopení jednotlivých principů relační algebry než její bezchybnou interpretaci. Z těchto důvodů je možné, že ve výsledném programu bude relační algebra upravena pro větší přehlednost výrazů a jednodušší pochopení jejích principů. Tímto by ale neměly být ovlivněny samotné možnosti relační algebry v programu.

Minimální množinu operací relační algebry tvoří operace sjednocení, průnik, kartézský součin, rozdíl, selekce a projekce. Pro přehlednost jsem se ještě rozhodl implementovat operaci obecného a přirozeného spojení. Všechny tyto operace podrobně popíšu a pokusím se je implementovat ve svém programu.

3.1. Pojem relační schéma, relace

Pro pochopení jednotlivých operací relační algebry je třeba aspoň stručně seznámit čtenáře s pojmem relační schéma a relace. Dle definice:

> *Relační schéma R je výraz tvaru R(A, f), kde R je jméno schématu, a = {A_1, A_2,…, A_n} je konečná množina jmen atributů, f je zobrazení přiřazující každému jménu atributu Ai neprázdnou množinu (obor hodnot atributu), kterou nazýváme doménou atributu D_i, tedy $f(A_i) = D_i$*

Výše zmíněná definice je uvedena jen pro úplnost. Z pohledu pochopení operací relační algebry nám stačí považovat relační schéma za atributy tabulky, určující její název a hlavičku (jednotlivé sloupce). Je nutno si uvědomit, že z pohledu jedné databáze musí být každá relace svým jménem unikátní.

> *Relace R s relačním schématem R je konečná podmnožina kartézského součinu domén Di, příslušejících jednotlivým atributům Ai, tedy $R \subset D_1 \times D_2 \times \ldots \times D_n$. Číslo n nazýváme stupněm relace, o relaci R říkáme, že je typu R nebo že je instancí relačního schématu R.*

Relaci je dle uvedené definice výhodné znázorňovat jako dvourozměrnou tabulku, kde každý řádek odpovídá jedné entitě (objektu z reálného světa) a každý sloupec jednomu atributu tabulky. Jména atributů musí být v rámci relace navzájem různá. Často se tedy v relačním modelu místo pojmu relace používá pojem tabulka.

Přestože je tato problematika mnohem složitější, v rámci této práce nám výše uvedený popis zcela postačí. Tyto definice jsou z [2].

3.2. O relační algebře

Pojem relační algebra se poprvé objevil v r. 1970 v článku „A relations model of data for large Sharp data banks“ dr. E. F. Codda, pracovníka firmy IBM, v časopise Communications of ACM. Relační algebra je prostředkem pro formulaci dotazů nad RMD (relačním modelem dat) a tvoří teoretický základ dotazovacích jazyků. Relační algebra už dle svého názvu pracuje s relacemi – operátory relační algebry se aplikují na relace a výsledkem jsou opět relace. Protože relace jsou množiny, je zřejmé, že velmi přirozenými budou především množinové operace. Další používané operace jsou již specificky databázové.

Relační algebra často není v této podobě implementována v SŘBD. Přesto je její pochopení nutné pro správnou manipulaci s relacemi - složitější dotazy jazyka SQL mohou být bez znalosti relační algebry problematické.

3.3. Testovací tabulky pro zobrazení výsledků operací

Pro pochopení operací relační algebry definuji v této části tabulky, na které se budu odkazovat při vysvětlování jednotlivých operací.

Relační schéma: Učitel (ČU, jméno, funkce, plat)
Relace: Učitel={(U12, Čáp, docent, 4000), (U27, Holub, asistent, 3000), (U39, Kos, asistent, 3000), (U43, Orel, docent, 4000), (U53, Datel, doktorand, 1500)}

ČU	Jméno	Funkce	Plat
U12	Čáp	docent	4000
U27	Holub	asistent	3000
U39	Kos	asistent	3000
U43	Orel	docent	4000
U53	Datel	doktorand	1500

Tabulka č. 1 – Testovací tabulka Učitel

Tabulka Učitel obsahuje seznam učitelů – číslo učitele (osobní číslo), jméno, funkci a jejich současný plat.

Relační schéma: Předmět (ČU, Název, Povinnost, Počet kreditů, Zakončení)
Relace: Předmět ={(U12, Uživatelská rozhraní, V, 4, KlZap), (U43, Základy elektroniky, P, 6, ZaZk), (U43, Programovací jazyky I, PV, 4, ZaZk),(U39, Teorie zpracování dat, PV, 6, ZaZk), (U27, Tvorba aplikací pro mobilní zařízení I, V, 6, ZaZk), (U39, Správa Windows systémů, V, 4, KlZap)}

ČU	Název	Povinnost	Kredity	Zakončení
U12	Uživatelská rozhraní	V	4	KlZap
U43	Základy elektroniky	P	5	ZaZk
U43	Programovací jazyky I	PV	4	ZaZk
U39	Teorie zpracování dat	PV	6	ZaZk
U27	Tvorba aplikací pro mobilní zařízení I	V	6	ZaZk
U39	Správa Windows systémů	V	5	KlZap

Tabulka č. 2 – Testovací tabulka Předmět

Tabulka Předmět obsahuje seznam vyučovaných předmětů – číslo učitele, který předmět učí, název předmětu, povinnost, počet kreditů za vykonání předmětu a způsob zakončení předmětu. Z výše uvedeného je vidět, že jde o relační schémata v kardinalitě 1:N. Jeden učitel může učit více předmětů, ale také žádný.

3.4. Databázové operace

Databázovými operacemi rozumíme operace selekce, projekce a spojení. Podle některých zdrojů se přes 80 % dotazů řeší těmito databázovými operacemi – nazývají se jednoduché dotazy. Jednoduché dotazy se v dotazovacích jazycích snadno konstruují a metody optimalizace jsou pro ně propracovány nejvíce.

Selekce

Selekce (nebo také restrikce) je značena písmenem σ. Syntaxe selekce v relační algebře je:

$$\sigma_{podmínka}(tabulka)$$

Pokud bychom tedy chtěli napsat v relační algebře dotaz pro získání všech asistentů z tabulky Učitel (tabulka č. 1), vypadal by takto:

$$\sigma_{Funkce=asistent}(Učitel)$$

Výsledná relace nad zadaným dotazem bude vypadat takto:

ČU	Jméno	Funkce	Plat
U27	Holub	asistent	3000
U39	Kos	asistent	3000

Tabulka č. 3 – Výsledek selekce č. 1 nad tabulkou Učitel

Operace selekce tedy vytvoří relaci s týmž schématem a ponechá entity z původní relace, které splňují zadanou logickou podmínku. Podmínka je zadána Booleovským výrazem (pomocí logických spojek $\wedge$ (a), $\vee$ (nebo) a $\neg$ (neplatí že) atomických formulí majících tvar $t_1\ \theta\ t_2$ kde $\theta \in \{<, >, =, \leq, \geq, <>\}$ a t_1 je buď konstanta, nebo jméno atributu.

Další možností, jakým specifikovat podmínku selekce je použití závorek. Na následujícím příkladu chceme z tabulky Předmět vybrat ty předměty, které jsou buď povinné (P) nebo povinně volitelné (PV) a zároveň jsou za 4 nebo 5 kreditů. Výraz v relační algebře pro získání této relace by vypadal takto:

$$\sigma(Povinnost=P \vee Povinnost=PV) \wedge (Kredity=4 \vee Kredity=5)\ (Předmět)$$

Výsledek pro tento dotaz je vidět zde:

ČU	Název	Povinnost	Kredity	Zakončení
U43	Základy elektroniky	P	5	ZaZk
U43	Programovací jazyky I	PV	4	ZaZk

Tabulka č. 4 – Výsledek selekce č. 2 nad tabulkou Předmět

Pro zajímavost uvádím, že v původním konceptu relační algebry nebyla uvažována podmínka IS NULL, tedy informaci, že atribut konkrétní entity není vyplněn.

Projekce

Projekce je značena písmenem π. Syntaxe projekce v relační algebře je:

$$\pi_{sloupec}\ (tabulka)$$

Projekce R[C] relace se schématem R(A) na množinu atributů C, kde B je podmnožinou a vytvoří relaci se schématem B a relacemi, které vzniknou z původní relace odstraněním hodnot atributů A-B. Zatímco tedy operací selekce omezujeme počet řádků, projekcí omezujeme počet sloupců. Dotaz s relací by mohl tedy například znít: Zobraz ČU a jméno z tabulky Učitel.

$$\pi_{ČU,\ Jméno}\ (Učitel)$$

Výsledek této projekce zobrazuje tabulka č. 5.

ČU	Jméno
U12	Čáp
U27	Holub
U39	Kos
U43	Orel
U53	Datel

Tabulka č. 5 – Výsledek projekce nad tabulkou Učitel

Při kombinaci operací projekce a selekce je třeba mít na paměti, že operace selekce se provádí vždy první. Nepotřebujeme tedy nutně dělat projekci sloupců, nad kterými chceme provádět selekci.

Přirozené a obecné spojení

Spojení relací R a s se schématy R(A), resp. R(B). Operace vytvoří největší relaci se schématem A ∪ B a relacemi, jejichž projekce na atributy A je z relace R a projekce B je z relace S. Spojení značíme R θ S, případně R * S. Pokud uvažujeme vzorec R * S, pak prvky nové relace vznikají spojením z obou relací přes rovnost hodnot na maximální množině společných atributů. Tomuto spojení se říká také přirozené spojení. Chceme-li vykonat dotaz pro seznam názvů všech předmětů a učitelů, kteří je učí, vzorec v relační algebře by vypadal takto:

$$\pi_{ČU,\,Jméno}(Učitel)\ \theta\ \pi_{ČU,\,Název}(Předmět)$$

Dostávám se tedy postupně ke složitějším – skládaným výrazům. V tomto výrazu nejdříve proběhne projekce tabulky Učitel, jsou zobrazeny jen sloupce ČU a Jméno. Z tabulky Předmět zobrazíme pomocí projekce sloupce ČU a Název. Tyto dvě relace poté spojíme pomocí přirozeného spojení. Výsledek dotazu je uveden v tabulce č. 6.

ČU	Jméno	Název
U12	Čáp	Uživatelská rozhraní
U43	Orel	Základy elektroniky
U43	Orel	Programovací jazyky I
U39	Kos	Teorie zpracování dat
U27	Holub	Tvorba aplikací pro mobilní zařízení I
U39	Kos	Správa Windows systémů

Tabulka č. 6 – Výsledek spojení tabulek Učitel a Předmět

Jak je vidět, duplicitní vzorec byl při spojení odstraněn. Výsledkem je seznam předmětů a učitelů, včetně jejich osobního čísla, přes které bylo provedeno spojení.

Přirozené spojení zde bylo umožněno díky tomu, že v tabulkách existuje množina společných atributů, tedy v tabulkách existují sloupce, které se jmenují stejně. Spojení může probíhat i nad více atributy. Pokud v tabulkách není žádný společný sloupec a my přesto chceme provést spojení, značíme tuto operaci obecného spojení jako R [t1 θ t2] s kde θ ∈ {<, >, =, ≤, ≥, <> }, t1 ∈ R(A) a t2 ∈ R(B).

Zatímco jsem tedy u přirozeného spojení mluvil pouze o porovnání rovnosti jednotlivých atributů, obecné spojení nám umožňuje použít i operandy uvedené výše.

3.5. Množinové operace

Jedná se o kartézský součin, sjednocení, průnik a rozdíl. Až na kartézský součin jsou všechny tyto operace známy z matematiky, přesto se je pokusím přehledně popsat. Sjednocení, průnik a rozdíl mají smysl tehdy, je-li počet atributů jednotlivých relací stejný a jsou-li tyto atributy kompatibilní. Na příkladech bude zřetelnější, proč tyto podmínky musí být splněny. Kartézský součin žádná omezení nemá.

Sjednocení

Sjednocení prvků je značeno ∪. Jde o spojení dvou množin – výsledná relace bude obsahovat všechny prvky z obou relací. Protože se současnými tabulkami nejde udělat příliš dobrý příklad sjednocení, pro představu definuji další. Tabulku definuji takto:

Relační schéma: Zaměstnanec (ČZ, Jméno, Funkce, Plat)
Relace: Zaměstnanec = {(Z28, Čáp, spec. pracovník, 1000), (Z35, Havran, školník, 800), (Z39, Sýkorka, uklízečka, 700)}

ČZ	Jméno	Funkce	Plat
Z28	Čáp	spec. pracovník	1000
Z35	Havran	školník	800
Z39	Sýkorka	uklízečka	700

Tabulka č. 7 – Tabulka Zaměstnanec

Z dřívějších tabulek je zřejmé, že uvažujeme instituci, ve které probíhá vyučování. Tato instituce bude mít samozřejmě zaměstnány i pracovníky neakademické. Vytvořil jsem tedy tabulku zaměstnanců. Pokud bychom chtěli získat seznam všech akademických pracovníků i obyčejných zaměstnanců z obou tabulek, tradiční operace nám k tomu nevystačí – tabulky není přes co spojit. Následující vzorec relační algebry ukazuje způsob provedení dotazu:

$$\pi_{Jméno}(Učitel) \cup \pi_{Jméno}(Zaměstnanec)$$

Jak je zřejmé, provedeme výběr sloupce Jméno z tabulky Učitel a výběr sloupce Jméno z tabulky Zaměstnanec. Získanou tabulku vidíme zde:

Jméno
Čáp
Holub
Kos
Orel
Datel
Havran
Sýkorka

Tabulka č. 8 – Výsledek operace sjednocení tabulek Učitel a Zaměstnanec

Tento příklad demonstruje, že provádět operaci sjednocení nad různými atributy je sice teoreticky možné, ale případným spojením do jednoho sloupce např. seznamu pracovníků a seznamem jejich platů nedostaneme nic užitečného. Sjednocení relací s různým počtem atributů je, jak je nyní zřejmější, už zcela nerealizovatelné. Tyto podmínky platí, jak již bylo zmíněno, i pro následující operace průniku a rozdílu.

Pozorný čtenář si jistě všiml, že pan Čáp je uveden jak v tabulce učitelů, tak v tabulce zaměstnanců, tedy že má dvě zaměstnání. Ve výsledné tabulce je ale uveden jen jednou – platí, že všechny tři operace filtrují řádky, které jsou již jednou v tabulce uvedeny.

Průnik

Operace průniku je značena ∩. Jde o operaci, kdy ve výsledné relaci budou jen ty řádky, které jsou obsaženy v obou relacích. Typickým příkladem by na našem příkladu učitelů a předmětů bylo zjištění všech učitelů, kteří učí nějaký předmět. Opět musíme provést nejdříve projekci a poté průnik. Výraz v relační algebře by vypadal takto:

$$\pi_{ČU}(Učitel) \cap \pi_{ČU}(Předmět)$$

Získáváme tedy dvě relace o jednom sloupci, které obsahují čísla učitelů. Průnikem získáme jen ty řádky, které jsou obsaženy v obou relacích. Výsledná tabulka:

ČU
U12
U27
U39
U43

Tabulka č. 9 – Průnik tabulek Předmět a Učitel

Odečítání

Odečítání se značí známým operátorem – (minus). Slouží k odečítání relací o shodném počtu sloupců, přičemž chceme, aby byly odstraněny všechny řádky, které jsou shodné s řádky druhé relace. Už z důvodu, že zde užívám označení „první“ a „druhý“ může čtenář usoudit, že odečítání není komutativní – evidentně záleží na pořadí relací. Často používaným příkladem by byl dotaz na všechny učitele, kteří neučí žádný předmět (tedy opačný příklad než při průniku výše). Stačí tedy vzít sloupec ČU z tabulky Učitel a odečíst od ní všechny ČU z tabulky Předmět.

$$\pi_{ČU}(Učitel) - \pi_{ČU}(Předmět)$$

Výsledkem této tabulky by byl pouze jeden řádek:

ČU
U53

Tabulka č. 10 – Odečtení tabulek Učitel a Předmět

Rychlým nahlédnutím do definicí tabulek výše zjistíme, že učitel s číslem U53, pan Datel, skutečně neučí žádný předmět. Můžeme položit ještě jeden podobný dotaz, ale v opačném pořadí, tedy:

$$\pi_{ČU}(Předmět) - \pi_{ČU}(Učitel)$$

Nepříliš šikovným výsledkem tohoto výrazu by byl seznam prázdných ČU (za předpokladu, že uvažujeme, že neučený předmět má atribut ČU vyplněn jako null). Tímto tedy můžeme zjistit aspoň počet předmětů, které nejsou učeny. Pokud žádné takové předměty nejsou, jako v našem případě, výsledkem výše popsaného dotazu bude prázdná množina.

Kartézský součin

Kartézský (direktní) součin je značen znakem ×. Pomocí něj získáme nad dvěma relacemi relaci, která bude obsahovat všechny uspořádané dvojice, kde první položka bude z první relace a druhá z relace druhé a výsledná relace obsahuje všechny kombinace těchto položek. Uvedu na příkladu, kdy bychom chtěli z nějakého důvodu zjistit všechny možné kombinace předmětů a učitelů. Výraz relační alebry pro získání takovéto relace by vypadal takto:

$$\pi_{Jméno,\ Název}\ (Předmět \times Učitel)$$

Pro úsporu místa zobrazím jen prvních několik řádků z 25 výsledné relace:

Jméno	Název
Čáp	Uživatelská rozhraní
Čáp	Základy elektroniky
Čáp	Programovací jazyky I
Čáp	Teorie zpracování dat
Čáp	Tvorba aplikací pro mobilní zařízení I
Čáp	Správa Windows systémů
Holub	Uživatelská rozhraní
Holub	Základy elektroniky
…	…

Tabulka č. 11 – Část výsledku kartézského součinu nad tabulkami Učitel a Předmět

Z tohoto příkladu můžeme usoudit, že kartézský součin je komutativní a nezáleží ani na počtu sloupců jednotlivých relací.

3.6. Další operace relační algebry

Jak jsem uvedl v úvodu kapitoly, nepopisoval jsem všechny operace relační algebry, ale téměř pouze ty tvořící minimální množinu operací. Dalšími operacemi relační algebry jsou přejmenování, levé a pravé polospojení, levé a pravé vnější spojení, vnější spojení a dělení. Více informací o těchto operacích a relační algebře obecně je možno najít v [1], [2] a [3].

4. SQL

V této části představím jazyk SQL a jeho syntaxi pro operace relační algebry uvedené v předchozí kapitole. SQL se v různých SŘBD může velice lišit a některé jeho funkce nemusí být ani implementovány. Není v rámci této práce podrobně popisovat rozdíly mezi syntaxí SQL jednotlivých SŘBD, vyberu si tedy pro popis syntaxe jedny z nejpoužívanějších, se kterými se jsem se již setkal a ze kterých v další části vyberu konkrétní SŘBD, se kterým bude můj program pracovat. Vybral jsem tyto:

- Oracle database
- Microsoft SQL Server
- MySQL

Pokud tedy dojde ke konfliktu syntaxe SQL v těchto SŘBD, pokusím se na něj upozornit a popsat řešení operací pro tyto tři vybrané SŘBD.

4.1. O SQL

SQL (structured query language - strukturovaný dotazovací jazyk) je standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích. V dnešních relačních SŘBD je jako uživatelský jazyk určen právě SQL. Jeho počátky ale sahají až do r. 1974 (původní název byl Sequel), kdy se zaměřoval ještě primárně na dotazovací část.

Kvůli jeho velice živelnému vývoji v první polovině 80. let vyvstala nutnost tento jazyk standardizovat – k tomu došlo v roce 1986. Poslední standard je z roku 2008. Popis rozdílů mezi jednotlivými standardy ale přesahuje rámec této práce.

SQL je více, než jen dotazovací jazyk. Dá se pomocí něj i data definovat a provádět aktualizace dat – tím se liší od relační algebry. Další možností je třeba definování uživatelských práv přístupu k relacím či definování indexů pro rychlejší přístup k prvkům relací.

4.2. Operace SQL v programu

SQL budu v prvé řadě používat ve svém programu jako jazyk dotazovací – uživatel vytvoří výraz v relační algebře a bude ho chtít přeložit. V následujících podkapitolách tedy proberu jednotlivé operace relační algebry probrané v předchozí kapitole a přiřadím jim jejich interpretaci v jazyce SQL.

Budu uvažovat také SQL příkazy, pomocí kterých uživatel bude moct definovat tabulku nebo vložit či vymazat data. Pro ukládání jednotlivých dotazů budu muset také ukládat jednotlivé části vzorce a zpětně je číst.

4.3. Databázové operace

V této části se budu přímo odkazovat na databázové operace z části relační algebra a ukážu syntaxi všech uvedených příkladů v SQL.

Selekce

Příkaz pro selekci se v SQL tvoří za klauzulí WHERE. Za ní specifikujeme všechny podmínky, které mají řádky výsledné selekce splňovat. Příklad dotazu bude vypadat stejně jako v relační algebře – chceme získat seznam všech asistentů z tabulky Učitel z tabulky č. 1. Tento dotaz v jazyce SQL by tedy vypadal takto:

```
SELECT *
FROM Učitel
WHERE Funkce = 'asistent';
```

Výsledek dotazu vidíme v tabulce č. 3. Stejně jako v relační algebře i v SQL můžeme samozřejmě podmínky libovolně spojovat. Klíčová slova pro spojení jsou AND (a), OR (nebo) a NOT (neplatí že). Pro spojování platí všechny podmínky popsané již v části zabývající se relační algebrou.

Ukažme si tedy příklad druhý se spojením podmínek - z tabulky Předmět chceme vybrat ty předměty, které jsou buď povinné (P) nebo povinně volitelné (PV) a zároveň jsou za 4 nebo 5 kreditů.

```
SELECT *
FROM Předmět
WHERE (Povinnost='P' OR Povinnost='PV')
AND (Kredity=4 OR Kredity=5);
```

Výsledek dotazu vidíme v tabulce č. 4.

Projekce

Jak je uvedeno v předchozí kapitole, projekcí definujeme sloupce, které chceme zobrazit ve výsledné relaci. V SQL definujeme projekci za klíčovým slovem SELECT a jednotlivé projekce oddělujeme čárkou. Chceme-li zobrazit všechny sloupce, použijeme výraz SELECT * (hvězdička). Spojujeme-li například dvě tabulky a chceme zobrazit z jedné všechny sloupce a z druhé jen jeden, použijeme tečkovou notaci:

```
SELECT tab1.*, tab2.sloupec
FROM tab1, tab2
```

Zobrazení ČU a jména z tabulky učitel by tedy v SQL vypadalo takto:

```
SELECT ČU, Jméno
FROM Učitel;
```

Výsledek takovéhoto dotazu je vidět v tabulce č. 5. Stejně jako v relační algebře i zde platí, že operace selekce se provádí před operací projekce.

Přirozené a obecné spojení

Spojování tabulek definujeme za klauzulí FROM. Klíčovými slovy pro tyto operace je NATURAL JOIN pro přirozené spojení a JOIN pro spojení obecné. Zde se dostáváme k problému velice různých možností jednotlivých SŘBD. MSSQL (Microsoft SQL) například nepodporuje operaci přirozeného spojení. Pro příklad přirozeného spojení opět použiji dotaz uvedený již v části s relační algebrou - Seznam názvů všech předmětů a učitelů, kteří je učí:

```
SELECT ČU, Jméno, Název
FROM Učitel
NATURAL JOIN Předmět;
```

Výsledek tohoto dotazu je zobrazen v tabulce č. 6. Syntaxe obecného spojení je poněkud složitější – je třeba uvést nejen, jaké tabulky chceme spojovat, ale také přes jaké atributy tabulek. Spojovací atributy se uvádí za klauzulí ON. Použijeme-li opět příklad uvedený výše, tentokrát přepsaný do podoby obecného spojení, dotaz by vypadal takto:

```
SELECT ČU, Jméno, Název
FROM Učitel
JOIN Předmět
ON Učitel.ČU=Předmět.ČU;
```

Je nutno použít tečkovou notaci z důvodu stejných názvů sloupců. Jak je již uvedeno v části s relační algebrou, můžeme použít i jiné operandy pro porovnání atributů, než je rovná se.

4.4. Množinové operace

V této části se budu přímo odkazovat na množinové operace z kapitoly relační algebra a ukážu syntaxi všech uvedených příkladů v SQL. Syntaxe pro jednotlivé SŘBD se zde velice liší, pokusím se tedy popsat řešení pro všechny tři zvolené SŘBD.

Sjednocení

Operaci sjednocení realizujeme pomocí klíčového slova UNION. Tato operace je definována ve všech třech SŘBD. Platí, že operaci sjednocení provádíme nad celými SELECT klauzulemi. Zvolíme-li opět příklad sjednocení z relační algebry, pak sjednocení tabulek Učitel a Zaměstnanec zapíšeme v SQL tímto způsobem:

```
SELECT Jméno
FROM Učitel
UNION
SELECT Jméno
FROM Zaměstnanec;
```

Výsledek dotazu si můžeme prohlédnout v tabulce č. 8.

Průnik

U průniku se syntaxe jednotlivých SŘBD liší. V Oracle a MSSQL by vypadal dotaz zjištění všech učitelů, kteří učí nějaký předmět takto:

```
SELECT ČU FROM Učitel
INTERSECT
SELECT ČU FROM Předmět;
```

V MySQL ale podobně jednoduché řešení není, průnik zde nemá vlastní klíčové slovo. Přesto je ale ale možné průnik udělat pomocí spojení takto:

```
SELECT DISTINCT Učitel.ČU
FROM Učitel
JOIN Předmět
ON Učitel.ČU=Předmět.ČU;
```

Přestože výsledek těchto dotazů (viz tabulka 9) je stejný, vidíme, že v MySQL je syntaxe dosti složitější. Toto se pokusím zahrnout do části výběru konkrétního SŘBD v další kapitole.

Odečítání

V MySQL opět není definováno jako klíčové slovo. V Oracle dosáhneme odečtení dvou relací klíčovým slovem MINUS, v MSSQL použijeme EXCEPT. Dotaz na seznam všech učitelů, kteří neučí žádný předmět by vypadal takto:

```
SELECT ČU
FROM Učitel
MINUS
SELECT ČU
FROM Předmět;
```

V MySQL bychom museli opět použít řešení pomocí levého spojení, poněkud složitější, přesto ale funkční:

```
SELECT Učitel.ČU
FROM Učitel
LEFT JOIN Předmět
ON Učitel.ČU=Předmět.ČU;
```

Výsledek pro oba dotazy vidíme vidíme v tabulce č. 10. Pokud bychom chtěli získat seznam (počet) předmětů, které nejsou učeny, pro Oracle a MSSQL nám stačí SELECT klauzule vyměnit. Pro řešení v MySQL musíme vyměnit nejen tabulky, ale také změnit projekci takto:

```
SELECT Předmět.ČU
FROM Předmět
LEFT JOIN Učitel
ON Předmět.ČU=Učitel.ČU;
```

Kartézský součin

Kartézský součin definujeme za klauzulí FROM. Tabulky, mezi kterými chceme kartézský součin provést, oddělíme jednoduše čárkou. Pro získání všech kombinací možností učitelů a předmětů by dotaz vypadal takto:

```
SELECT Jméno, Název
FROM Předmět, Učitel;
```

Výsledná relace tohoto dotazu je zobrazena v tabulce 11.

5. Výběr SŘBD

V této části se pokusím popsat výběr konkrétního SŘBD, nad kterým bude výsledná aplikace pracovat. Jak jsem uvedl již v předchozí kapitole, uvažoval jsem tyto SŘBD:

- Oracle database
- Microsoft SQL Server
- MySQL

Se všemi třemi jsem se již setkal a pracoval, osobně mi, co se týče možností a rychlosti, nejvíce vyhovovala Oracle database, na rozdíl od Microsoft SQL Serveru, který mi nepřišel zdaleka tak přehledný a funkční. Protože ale jedním z mých cílů byla snadná dostupnost a snadnost zavedení výsledného programu, chtěl jsem nejdříve probrat možnost nekomerční, tedy MySQL. Možnost stáhnutí a instalace přes průvodce celého MySQL serveru mi v tomto případě přišla ideální.

Dalším důvodem proč jsem původně uvažoval MySQL bylo, že při seznámení se s několika hotovými systémy pro překlad relační algebry tyto pracovaly právě s Oracle database, chtěl jsem se tedy pokusit vytvořit překladač, který nepracuje nad stejným SŘBD, jako již vytvořené překladače.

Podrobným studiem jednotlivých příkazů relační algebry a těchto SŘBD jsem ale zjistil, že množinové operace sjednocení, průniku a odečítání nejsou v MySQL přímo naimplementovány. Jejich realizace je možná, jak je zmíněno v předchozí kapitole, ale výsledný kód SQL je mnohokrát složitější a méně přehledný.

Tento složitější kód však může mít, jak se dá zjistit pomocí možnosti Explain Plan v Oracle SQL Developeru menší nároky na vyřešení dotazu, jak ukazuje následující obrázek:

OPERATION	OBJECT_NAME	OPTIONS	COST
SELECT STATEMENT			7
MINUS			
SORT		UNIQUE	3
INDEX	SYS_C00506445	FAST FULL SCAN	2
SORT		UNIQUE	4
TABLE ACCESS	TESTOVACI_REZERVACE	FULL	3

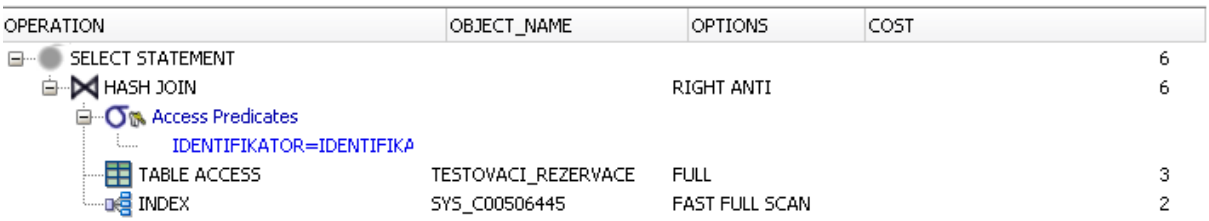
Obrázek č. 1 – Cena dotazu obsahujícího klíčové slovo minus

Obrázek výše ukazuje relativní cenu tohoto dotazu:

```
SELECT id FROM Uzivatel
MINUS
SELECT id_uzivatel FROM Rezervace;
```

Následující obrázek ukazuje relativní cenu dotazu:

```
SELECT id
FROM Uzivatel
LEFT JOIN Rezervace
ON Uzivatel.id=Rezervace.id_uzivatel
WHERE Rezervace.id_uzivatel IS NULL;
```



Obrázek č. 2 – Cena dotazu přes LEFT JOIN

První i druhý dotaz si výslednou relací odpovídají. Přesto se ceny dotazů mírně liší. Menší cena složitějšího dotazu byla další motivací, proč uvažovat v programu použití MySQL.

Protože ale vytváření složitějšího dotazu kladlo velké nároky i na samotný kód programu, po pokusném překladu několika složitějších výrazů jsem se rozhodl, že použiji také Oracle database, která mi umožní vytvářet výsledné SQL dotazy mnohem přehlednější.

6. Datová analýza

V této části bych rád představil datovou analýzu. Navrhnu v ní způsob, jakým budou data ukládána, jaký budou mít formát a v jakém budou vztahu. V této části se už tedy pohybujeme skutečně u tabulek, nezaměňovat tedy s relacemi v relační algebře.

6.1. Pojmy v datové analýze

Následuje popis pojmů, se kterými budu v datové analýze pracovat. Podrobnosti o těchto pojmech lze najít v [1], [2] a [4].

Databázová tabulka

Databázová tabulka slouží k přímému uložení dat do databáze. Můžeme si ji představit jako dvourozměrnou tabulku, která má jistý počet sloupců (atributů) a řádků (záznamů). Tabulka je definována svým unikátním názvem, seznamem atributů, údajem o primárních a cizích klíčích a případně pravidly referenční integrity. V ERD (Entity-relationship diagram) ji budu zobrazovat jako obdélník, kde v prvním jeho řádku je název tabulky a pod ním seznam jednotlivých atributů.

Primární klíč

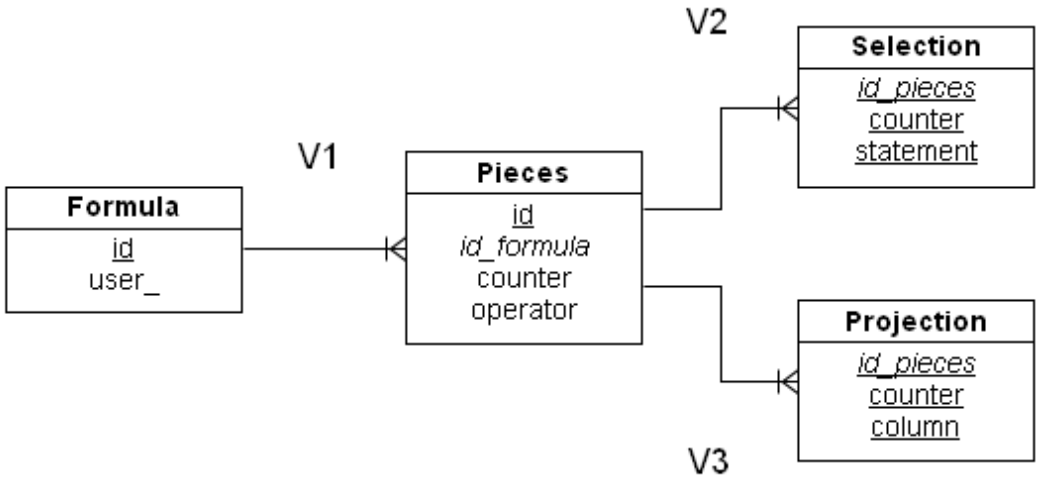
Primární klíč je pole nebo kombinace polí v záznamu, které záznam jednoznačně identifikují. Z tohoto vyplývá, že žádné pole primárního klíče nesmí být nevyplněno a žádná dvě pole primárního klíče nesmí obsahovat stejnou hodnotu. V lineárním zápisu typů entit (viz níže) ho budu značit podtrženě.

Cizí klíč

Jde o pole, které označuje vztah mezi řádky tabulek, ve kterých se hodnoty primárního a cizího klíče rovnají. Cizí klíč tedy odkazuje na hodnotu primárního klíče. Z tohoto vyplývá, že hodnota pole cizího klíče nemůže existovat bez rovnající se hodnoty pole primárního klíče. Při mazání dat s primárním klíčem, na který je odkazováno v cizím klíči, musíme nejdřív smazat řádky podřízené tabulky, která na tento primární klíč odkazuje. V lineárním zápisu typů entit ho budu označovat kurzívou.

6.2. ER diagram

Označován jako ERD slouží pro abstraktní a konceptuální znázornění dat a vazeb mezi nimi. Diagram by měl přehledně zobrazovat jednotlivé entitní typy, jejich atributy a typy vztahů mezi nimi. V tomto případě by tedy čtenář měl získat přehled o tom, jakým způsobem budou definovány tabulky pro ukládání výrazů relační algebry v programu. S ERD úzce souvisí lineární zápis typů entit, který nám zobrazí primární a cizí klíče jednotlivých tabulek. Pro popis vazeb použijeme lineární zápis typů vazeb, který by měl čtenáři přiblížit vztahy mezi entitními typy.



Obrázek č. 3 – ER diagram popisující tabulky pro ukládání výrazů relační algebry

6.3. Lineární zápis typů entit

Formula(id, user_)
Pieces(id, *id_formula*, counter, operator)
Selection(*id_pieces*, counter, selection)
Projection(*id_pieces*, counter, column_name)

6.4. Lineární zápis typů vazeb

V1 : Je_složena_z(Formula, Pieces) 1:N
V2 : Může_obsahovat(Pieces, Selection) 1:N
V3 : Může_obsahovat(Pieces, Projection) 1:N

6.5. Datový slovník

Datový slovník nám umožňuje přehledně zobrazit jednotlivé tabulky, jejich atributy a vlastnosti těchto atributů.

Entita č. 1 - Formula

Tato tabulka obsahuje seznam všech výrazů uložených v databázi. Nad ní můžeme zjistit, který uživatel má kolik uložených výrazů.

Název	Datový typ	Velikost	Klíč	Null	Index	Popis
id	number	22	Ano	Ne	Ano	Id uživatele
id_formula	number	22	Ne	Ano	Ano	Cizí klíč z tabulky Formula
counter	number	22	Ne	Ano	Ne	Určuje pořadí jednotlivých částí vzorce
operator	varchar	250	Ne	Ano	Ne	Jméno tabulky nebo operand

Tabulka č. 12 – Datový slovník entity č. 1 - Formula

Entita č. 2 – Pieces

Tabulka Pieces obsahuje části jednotlivých výrazů. Ve své podstatě každý řádek této tabulky odpovídá jedné části výrazu v aplikaci. Cizí klíč id_formula odkazuje na vzorec, ke které daná část vzorce patří. Dále obsahuje tzv. counter, který nám určuje pořadí jednotlivých kousků vzorce.

Název	Datový typ	Velikost	Klíč	Null	Index	Popis
id	number	x	Ano	Ne	Ano	Id uživatele
user_	varchar	250	Ne	Ano	Ne	Přihlašovací jméno uživatele

Tabulka č. 13 – Datový slovník entity č. 2 - Pieces

Entita č. 3 – Selection

Tato tabulka obsahuje seznam selekcí prováděné nad danou tabulkou včetně jednotlivých operandů booleovské algebry nebo závorek. Id_pieces označuje, ke které části výrazu daná selekce patří. Counter opět označuje pořadí těchto selekcí.

Název	Datový typ	Velikost	Klíč	Null	Index	Popis
id_pieces	number	22	Ne	Ano	Ano	Cizí klíč z tabulky Pieces
counter	number	22	Ne	Ano	Ne	Určuje pořadí jednotlivých částí selekce
selection	varchar	250	Ne	Ano	Ne	Část selekce nebo operand

Tabulka č. 14 – Datový slovník entity č. 3 - Selection

Entita č. 4 – Projection

Podobně jako tabulka Selection se vztahuje k jednotlivým částem vzorce. Obsahuje seznam projekcí nad danou části výrazu relační algebry.

Název	Datový typ	Velikost	Klíč	Null	Index	Popis
id_pieces	number	2	Ne	Ano	Ano	Cizí klíč z tabulky Pieces
counter	number	2	Ne	Ano	Ne	Určuje pořadí jednotlivých sloupců
column_name	varchar	250	Ne	Ano	Ne	Název sloupce projekce

Tabulka č. 15 – Datový slovník entity č. 4 - Projection

6.6. Systémový katalog

V aplikaci bude nutné používat tabulky systémového katalogu. Jde o interní informace o databázi, které se vytváří automaticky dle zásahů uživatele do databáze. V aplikaci budu používat následující systémové tabulky:

- user_objects
- all_constraint
- all_cons_columns
- all_tab_columns

6.7. Popis používaných tabulek systémového katalogu

Zde popíšu stručně obsah jednotlivých tabulek systémového katalogu, které používám. Podrobnější informace, jakým způsobem a za jakým účelem přistupuji k jednotlivým tabulkám je možno najít v části Funkční analýza.

Tabulka informačního schématu č. 1 - user_objects

Tato tabulka obsahuje názvy všech objektů v databázi. Jednoduchou selekcí je možno získat všechny tabulky v databázi.

Tabulka informačního schématu č. 2 - all_constraints

Tabulka all_*constraints* popisuje jednotlivé omezení nad tabulkami. K těmto omezením patří například primární klíče. Selekcí v této tabulce tedy můžeme získat seznam primárních klíčů pro všechny tabulky.

Tabulka informačního schématu č. 3 – all_cons_columns

Tato tabulka popisuje přiřazení jednotlivých omezení nad tabulkami z tabulky *all_constraints* ke konkrétním sloupcům. Vyfiltrováním primárního klíče nad konkrétní tabulkou z tabulky informačního schématu *all_constraints* získáme jednoznačné označení primárního klíče v databázi. Pokud toto označení spojíme s tabulkou *all_cons_columns*, získáme informaci o názvu sloupce tabulky, obsahujícího primární klíč.

Tabulka informačního schématu č. 4 – all_tab_columns

Tabulka *all_tab_columns* obsahuje seznam všech sloupců tabulek v celé databázi, jejich datový typ a další. Selekcí záznamů dle názvu tabulky můžeme získat seznam všech sloupců dané tabulky.

7. Funkční analýza

V této části práce se budu zabývat vysvětlením jednotlivých funkcí programu pomocí DFD (data flow diagramů). Tato kapitola by měla umožnit čtenáři pochopit hlavní funkce programu a princip jejich realizace. Kompletní DFD navrhované aplikace je uvedeno jako příloha na bakalářské práce. Více o DFD je možno najít např. v [5].

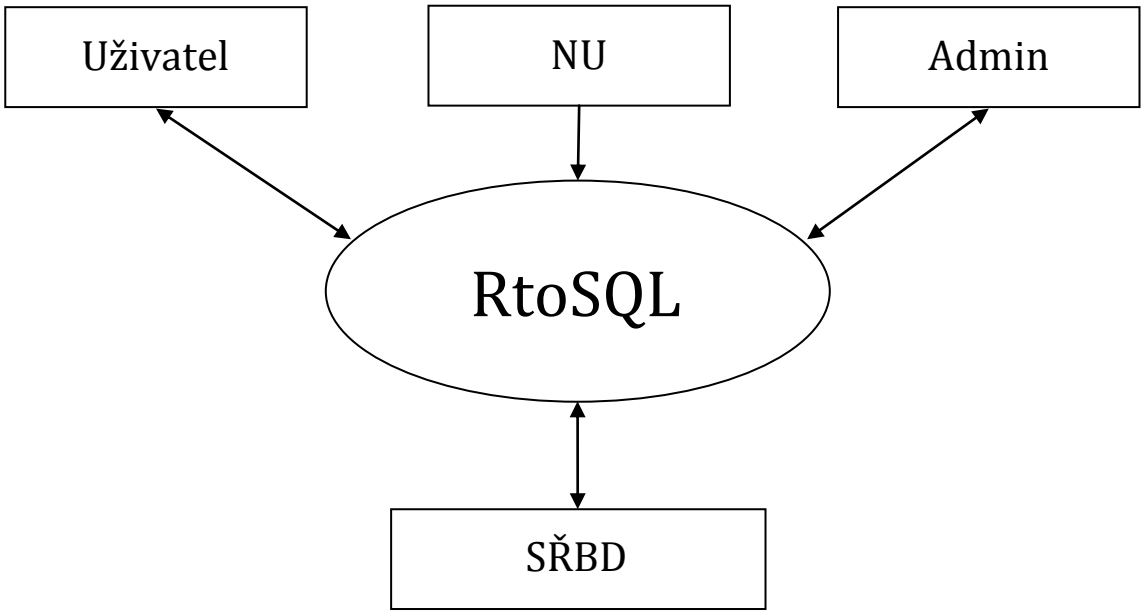
7.1. DFD

DFD je grafický prostředek pro návrh a zobrazení funkčního modelu systému. Zobrazuje jednotlivé procesy systému a datové toky mezi procesy, datovými úložišti a aktéry. DFD má být dostatečně jednoduchý pro pochopení a může sloužit i k upřesňování zadání. DFD rozdělíme na čtyři části pro snazší pochopení funkcí aplikace:

- Kontextový diagram
- 0. úroveň – rozdělení na moduly
- 1. úroveň
- Minispecifikace

7.2. Kontextový diagram

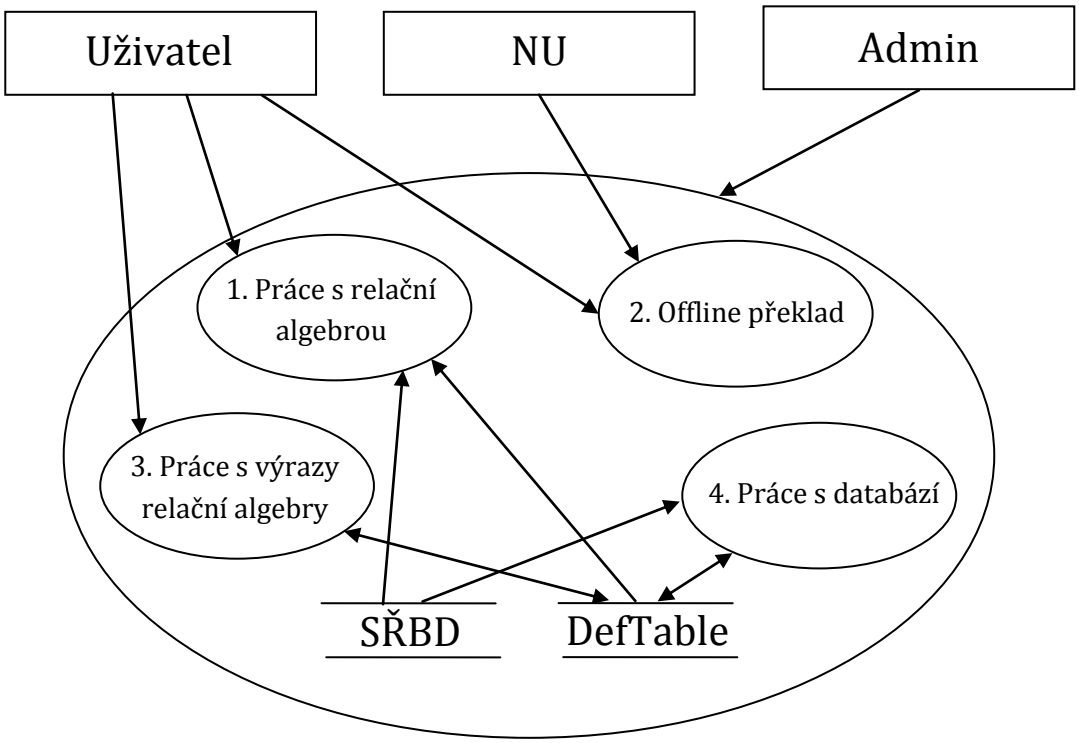
Kontextový diagram je nejvyšší úrovní pohledu na systém. Zobrazuje jednotlivé aktéry přistupující k systému, komunikaci se SŘBD a datové toky mezi nimi.



Obrázek č. 4 – DFD Kontextový diagram

7.3. 0. úroveň – rozdělení na moduly

V této úrovni vidíme hlavní části aplikace RtoSQL a role, které k daným částem mají přístup. Datastore SŘBD uvažujeme jako seznam tabulek systémového katalogu, který používám. DefTable je buď seznam tabulek, nad kterými voláme dotaz, konkrétní tabulka, ve které definujeme data, nebo tabulky používané pro uložení výrazů relační algebry. Toto rozvrhnutí jsem zvolil proto, že nejde o tradiční informační systém, tabulky volí dynamicky uživatel, tedy nelze uvažovat konkrétní názvy tabulek.

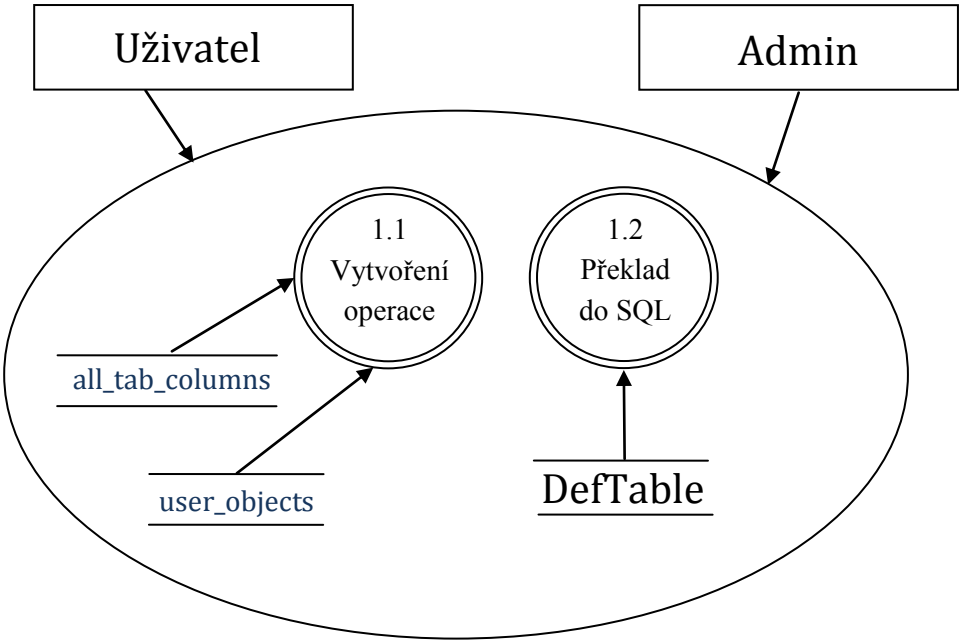


Obrázek č. 5 – DFD 0. úroveň

7.4. 1. úroveň – 1. Práce s relační algebrou

V této části se věnuji popisu funkcí pro práci s relační algebrou. Neuvažuju vytváření samotných výrazů, kdy ve své podstatě nedochází k datovým tokům mezi databází a programem. Návrh vytváření samotných výrazů je uveden až v části návrhu implementace. Uvedený datastore DefTable reprezentuje seznam tabulek, nad kterými je volán dynamicky vytvořený dotaz. Jako práci s relační algebrou tedy uvažuji:

- Vytvoření operace
- Překlad do SQL

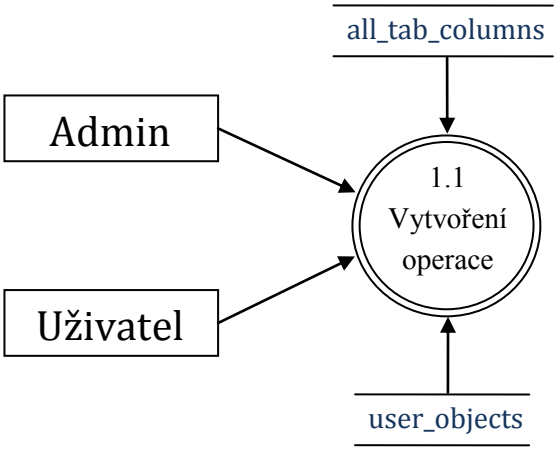


Obrázek č. 6 – DFD 1. úroveň 1. Práce s relační algebrou

7.5. Minispecifikace č. 1 – Práce s relační algebrou

V této části jsou podrobně rozepsány funkce nad částí č. 1 – Práce s relační algebrou

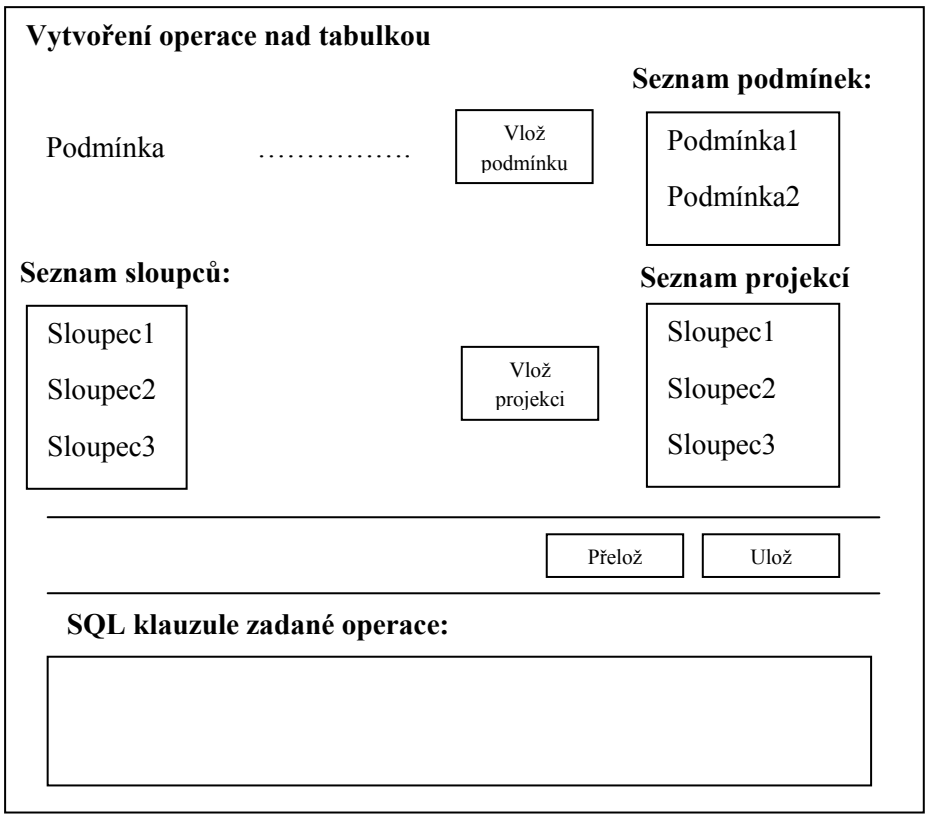
Vytvoření operace



Obrázek č. 7 – Minispecifikace 1.1

Algoritmus pro vytvoření operace:

1) Z tabulky user_objects načti seznam všech dostupných tabulek.
2) Zobraz formulář RtoSQL se seznamem tabulek.
3) Uživatel vybere tabulku, nad kterou chce vytvářet operace a vybere možnost na vytvoření operace nad tabulkou.
4) Získej seznam sloupců a datových typů vybrané tabulky z tabulky all_tab_columns a zobraz je do formuláře FormOperations.
5) Uživatel nastaví nad tabulkou požadované projekce a selekce, převede operaci do relační algebry.
6) Pokusí-li se uživatel vložit nepovolený znak (středník, uvozovky) do selekce, zobraz chybu „podmínka nesmí obsahovat středník a uvozovky“ a jdi na bod 5.
7) Zkontroluj selekci nastavenou uživatelem.
8) Vyskytne-li se chyba u nastavení podmínek selekce, zobraz chybu „operandy nesprávně nastaveny“ a jdi na bod 5.
9) Je-li nastavení selekce v pořádku, převeď danou operaci nad tabulkou do SQL a zobraz uživateli.
10) Uživatel vloží operaci nad tabulkou do tvůrce výrazů.
11) Zobraz dialogové okno pro setrvání ve stávajícím okně s možnostmi ano/ne .
12) Označí-li uživatel možnost ano, návrat do tvůrce výrazů.
13) Označí-li uživatel možnost ne, vytváří další operaci nad stejnou tabulkou.

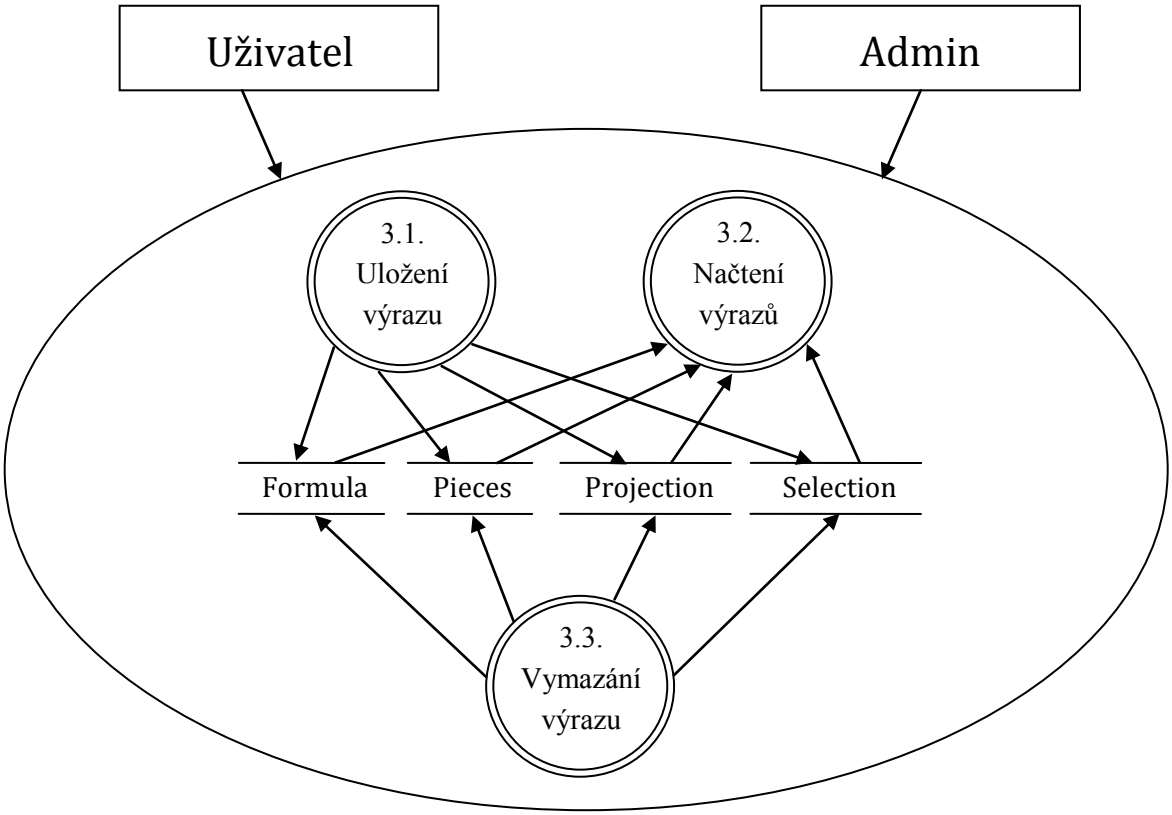


Obrázek č. 8 – Minispecifikace 1.1 č. 2

7.6. 1. úroveň – 3. Práce se vzorci relační algebry

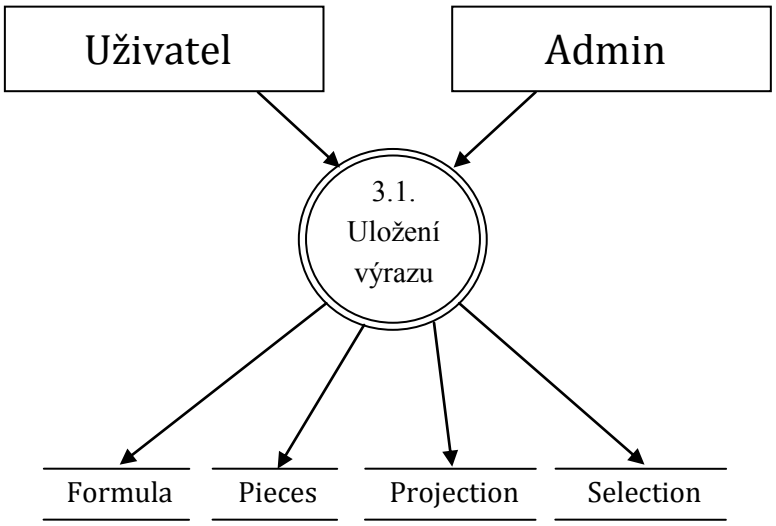
V této části popíšu, jakým způsobem mohou uživatelé pracovat s vytvořenými vzorci v relační algebře. Je umožněno vytvořené vzorce:

- Ukládat
- Načítat
- Mazat



Obrázek č. 9 – DFD 1. úroveň 3. Práce s výrazy relační algebry

Uložení výrazu



Obrázek č. 10 – Minispecifikace 3.1

Algoritmus uložení výrazu:

1) Zobraz formulář RtoSQL.
2) Uživatel vytvoří libovolný výraz relační algebry a zvolí možnost uložení výrazu.
3) Do tabulky Formula ulož login uživatele získaný z přihlášení k SŘBD v programu a jedinečné číslo vzorce.
4) Projdi jednotlivé operace seznamu ve formuláři RtoSQL.
5) Je-li konkrétní operace operandem relační algebry nebo závorkou, ulož ji do tabulky Pieces s jedinečným číslem vzorce a pořadím uložení daného operandu.
6) Je-li konkrétní operace operací nad tabulkou, ulož název tabulky do tabulky Pieces
7) Zjisti operace projekce a selekce pro danou tabulku.
8) Vlož do tabulky Projection jedinečné číslo operace z tabulky Pieces, název sloupce a pořadí dané projekce.
9) Vlož do tabulky Selection jedinečné číslo operace z tabulky Pieces, podmínku a pořadí dané selekce.
10) Vyskytne-li se chyba, zobraz chybu „Vzorec se nezdařilo uložit“, jdi na bod 2.

Načtení výrazů

1) Uživatel zvolí možnost pro zobrazení uložených výrazů relační algebry ve formulářovém okně RtoSQL.
2) Získej všechny jedinečné klíče výrazů z tabulky Formula, kde je login uživatele roven loginu právě přihlášeného uživatele, vytvoř seznam získaných výrazů.
3) Projdi jedinečné klíče všech výrazů a nad každým získej seznam operaci z tabulky Pieces, přidej do seznamu výrazů a seřaď dle čísla uložení.
4) Projdi všechny získané operace, zjisti jejich jedinečný klíč a získej všechny záznamy z tabulek Projection a Selection, ve kterých je jedinečný klíč operace roven cizímu klíči v tabulkách Projection a Selection, seřaď je podle čísla uložení.
5) Podle získaných informací vytvoř seznam uložených výrazů.
6) Vyskytne-li se někde během procesu chyba na úrovni SŘBD, je tato chyba zobrazena uživateli
7) Uživatel vybere konkrétní vzorec, který požaduje do hlavního okna.

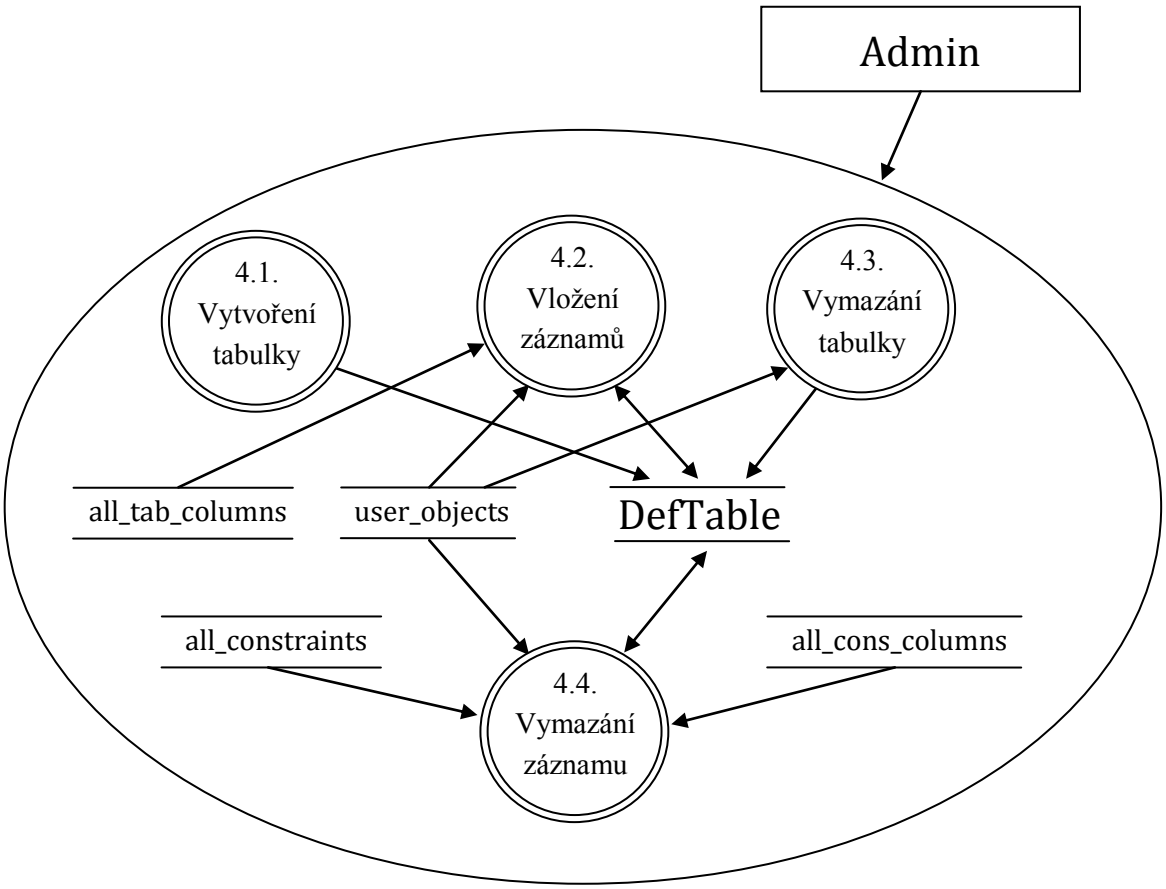
Smazání výrazu

1) Uživatel označí ve formuláři ListOfFormulas výraz, který chce smazat.
2) Ze seznamu výrazů je získáno jedinečné označení výrazu.
3) Z tabulky Pieces je získán seznam záznamů patřících k danému výrazu dle jedinečného označení výrazu.
4) Z tabulky Selection a Projection je získán seznam záznamů patřících k dané operaci nad tabulkou dle primárního klíče tabulky Pieces.
5) Vymaž získaný seznam záznamů z tabulky Projection.
6) Vymaž získaný seznam záznamů z tabulky Selection.
7) Vymaž získaný seznam záznamů z tabulky Pieces.
8) Vymaž záznam s jedinečným označením konkrétního výrazu, který uživatel chce vymazat.
9) Vyskytla-li se během provádění chyba, zobraz chybu na úrovni SŘBD uživateli, jdi na bod 1.

7.7. 1. úroveň – 4. Práce s databází

V této části se věnuji popisu funkcí pro práci s databází. Do této části by měl mít přístup pouze administrátor. Jednotlivé funkce pro práci s databází vidíme na DFD níže, jde o funkce:

- Vytvoření tabulky
- Vložení záznamů
- Vymazání tabulky
- Vymazání záznamu

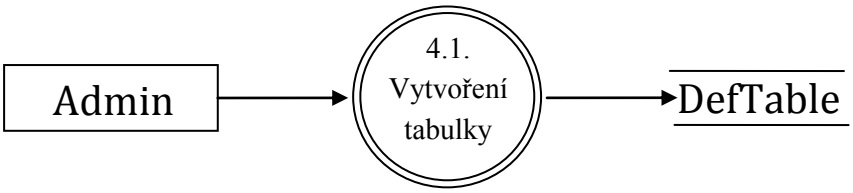


Obrázek č. 11 – DFD 1. úroveň 4. Práce s databází

7.8. Minispecifikace č. 4 – Práce s databází

V této části jsou podrobně rozepsány funkce nad částí č. 4 – Práce s databází.

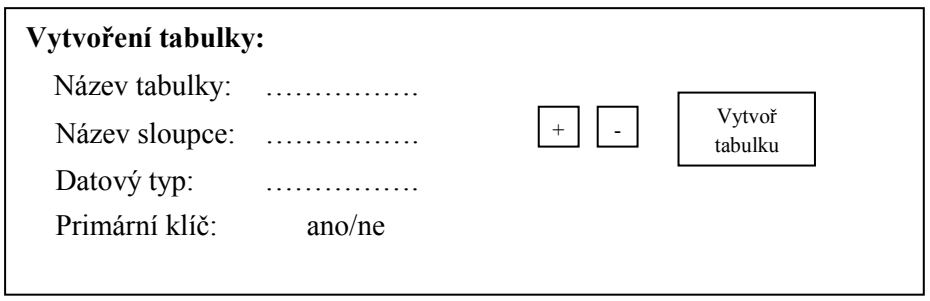
Vytvoření tabulky



Obrázek č. 12 – Minispecifikace 4.1

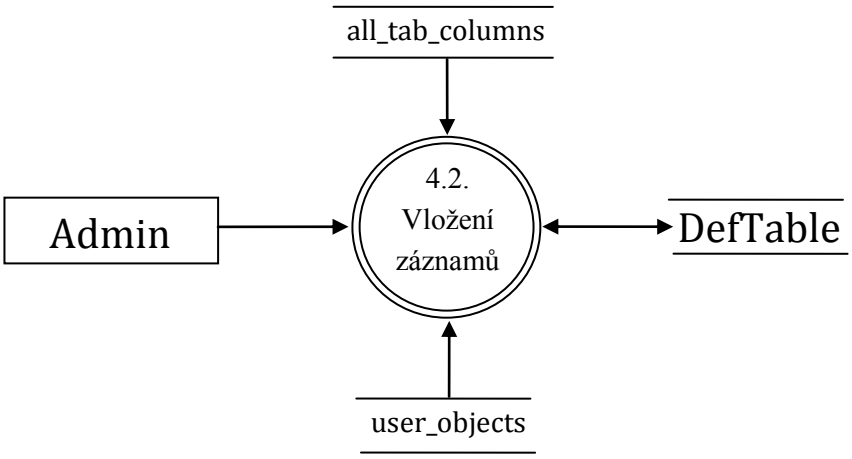
Algoritmus vytvoření tabulky:

1) Zobraz formulář FormCreateTable s nevyplněným názvem a sloupci tabulky.
2) Administrátor vyplní název tabulky, požadované sloupce a datové typy.
3) Zkontroluj, zda je vyplněn název tabulky.
4) Není-li vyplněn název tabulky, vypiš chybové hlášení „Není zadán název tabulky“ a jdi na bod 2.
5) Zkontroluj, zda jsou vyplněny názvy všech sloupců a jejich datových typů.
6) Pokud ne, vypiš chybové hlášení „Některý parametr tabulky není vyplněn“, jdi na bod 2.
7) Zkontroluj, zda je jeden ze sloupců označen jako primární klíč.
8) Není-li některý ze sloupců označen jako sloupec s primárním klíčem, vypiš chybové hlášení „Není označen primární klíč“, jdi na bod 2.
9) Pokud administrátor vyplnil vše správně, z vyplněných proměnných vygeneruj příkaz CREATE TABLE pro vytvoření tabulky a spusť příkaz nad databází.
10) Vyskytne-li se problém na úrovni SŘBD, zobraz chybu administrátorovi a jdi na bod 2.
11) Proběhne-li vytvoření tabulky korektně, zobraz hlášení „Tabulka úspěšně vytvořena“ následováno vygenerovaným kódem pro vytvoření tabulky.
12) Výsledkem je vytvoření tabulky DefTable dle zadaných parametrů.



Obrázek č. 13 – Minispecifikace 4.1 č. 2

Vložení záznamů



Obrázek č. 14 – Minispecifikace 4.2

Algoritmus vložení záznamů:

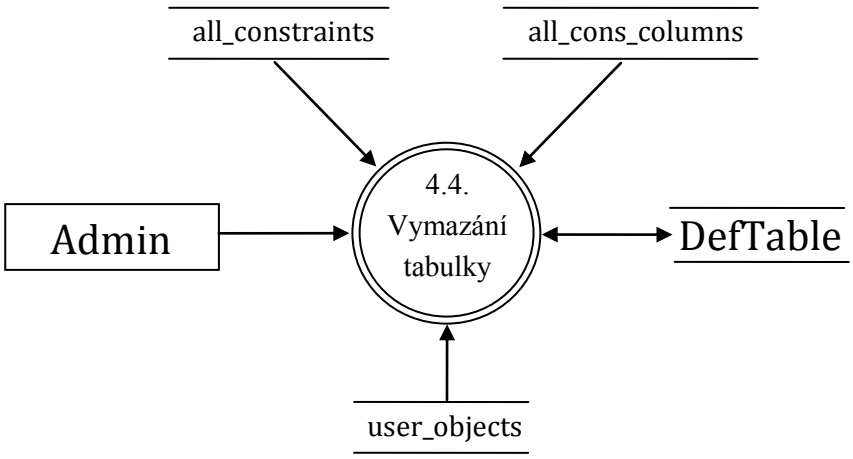
1) Z tabulky user_objects načti seznam všech dostupných tabulek.
2) Zobraz formulář FormAdministration se seznamem tabulek.
3) Administrátor vybere tabulku, do které chce vkládat data.
4) Načti obsah administrátorem vybrané tabulky ze SŘBD.
5) Zobraz data tabulky ve formuláři FormInsertIntoShowData.
6) Získej názvy jednotlivých sloupců a jejich datové typy administrátorem určené tabulky z all_tab_columns.
7) Zobraz formulář pro vkládání dat FormInsertInto a dynamicky ho vyplň poli podle počtu sloupců tabulky.
8) Administrátor vyplní libovolný počet řádků tabulky a vybere volbu vložení dat.
9) Je vytvořen seznam příkazů pro vkládání definovaných dat, není-li někde vyplněna hodnota, je do příkazu vložena hodnota null.
10) Vyskytne-li se na úrovni SŘBD chyba, je tato zobrazena administrátorovi, jdi na bod 8.
11) Proběhne-li vložení korektně, zobraz hlášení „Úspěšně vložen příkaz“ následovaný vygenerovaným příkazem.
12) Výsledkem je vložení zadaných dat do tabulky DefTable.
13) Aktualizuj formulář vyplněný daty tabulky DefTable.

3. Vymazání tabulky

Algoritmus vymazání tabulky:

1) Z tabulky user_objects načti seznam všech dostupných tabulek.
2) Administrátor vybere tabulku, kterou chce vymazat.
3) Zobraz dialogové okno s výzvou: „Skutečně vymazat tabulku?“, doplněnou názvem tabulky a možnostmi ano/ne.
4) Vybere-li uživatel možnost ne, jdi na bod 1.
5) Vybere-li uživatel možnost ano, je vygenerován kód pro vymazání tabulky a spuštěn nad databází.
6) Vyskytne-li se na úrovni SŘBD chyba, je tato zobrazena administrátorovi, jdi na bod 1.
7) Proběhne-li příkaz korektně, je tabulka úspěšně vymazána

4. Vymazání záznamu



Obrázek č. 15 – Minispecifikace 4.4

Algoritmus pro vymazání záznamu z tabulky:

1) Z tabulky user_objects získej seznam všech dostupných tabulek.
2) Administrátor vybere tabulku, ze které chce mazat záznamy.
3) Spojením tabulek all_constraints a all_cons_columns získej název sloupce obsahující primární klíč.
4) Neexistuje-li v tabulce primární klíč, zobraz chybové hlášení „Není určen primární klíč tabulky, není možno mazat data“, jdi na bod 1.
5) Administrátor označí záznam, který chce vymazat.
6) Zobraz dialogové okno „Skutečně chcete smazat tento záznam“ doplněný možnostmi ano/ne.
7) Vybere-li uživatel možnost ne, jdi na bod 5.
8) Vybere-li uživatel možnost ano, načti hodnotu buňky vybraného záznamu sloupce s označením primárního klíče.
9) Ze získané hodnoty vygeneruj příkaz DELETE FROM WHERE, kde podmínka vymazání bude atribut s primárním klíčem rovná se hodnota.
10) Vygenerovaný příkaz zavolej nad databází.
11) Vyskytne-li se na úrovni SŘBD chyba, zobraz chybu zobrazena administrátorovi, jdi na bod 5.
12) Nevyskytne-li se chyba, je záznam korektně vymazán.
13) Aktualizuj zobrazený formulář.

8. Návrh implementace

V této kapitole se pokusím navrhnout před samotnou implementací hlavní principy práce programu a popsat je čtenáři. Hlavní částí budou popis způsobu překladu výrazů a návrh uživatelského rozhraní.

8.1. Indexová analýza

Vytvoření DFD mi mělo pomoci rozhodnout, zda se často vyhledává podle jiných než již indexovaných atributů. V kapitole datové analýzy jsem indexy umístil pouze na primární a cizí klíče. Vzhledem k tomu, že se neočekává, že by jeden uživatel měl uloženo velké množství výrazů, rozhodl jsem, že není třeba vkládat další indexy kromě již zmíněných primárních a cizích klíčů. Při běžném provozu by se nemělo stát, že by velký počet výrazů nějakým způsobem negativně ovlivnil rychlost načítaní výrazů z tabulek.

8.2. Transakční analýza

V této části mám pomocí minispecifikací určit, zda bude třeba v aplikaci využít transakce. Z důvodu, že při ukládání a mazání výrazů pracujeme zároveň nad několika tabulkami, přičemž výraz nemá smysl za předpokladu, že se při ukládání některé z jeho části vyskytne chyba, realizuji ukládání a mazání pomocí transakcí. Více o transakcích v [4].

8.3. Relační algebra v programu

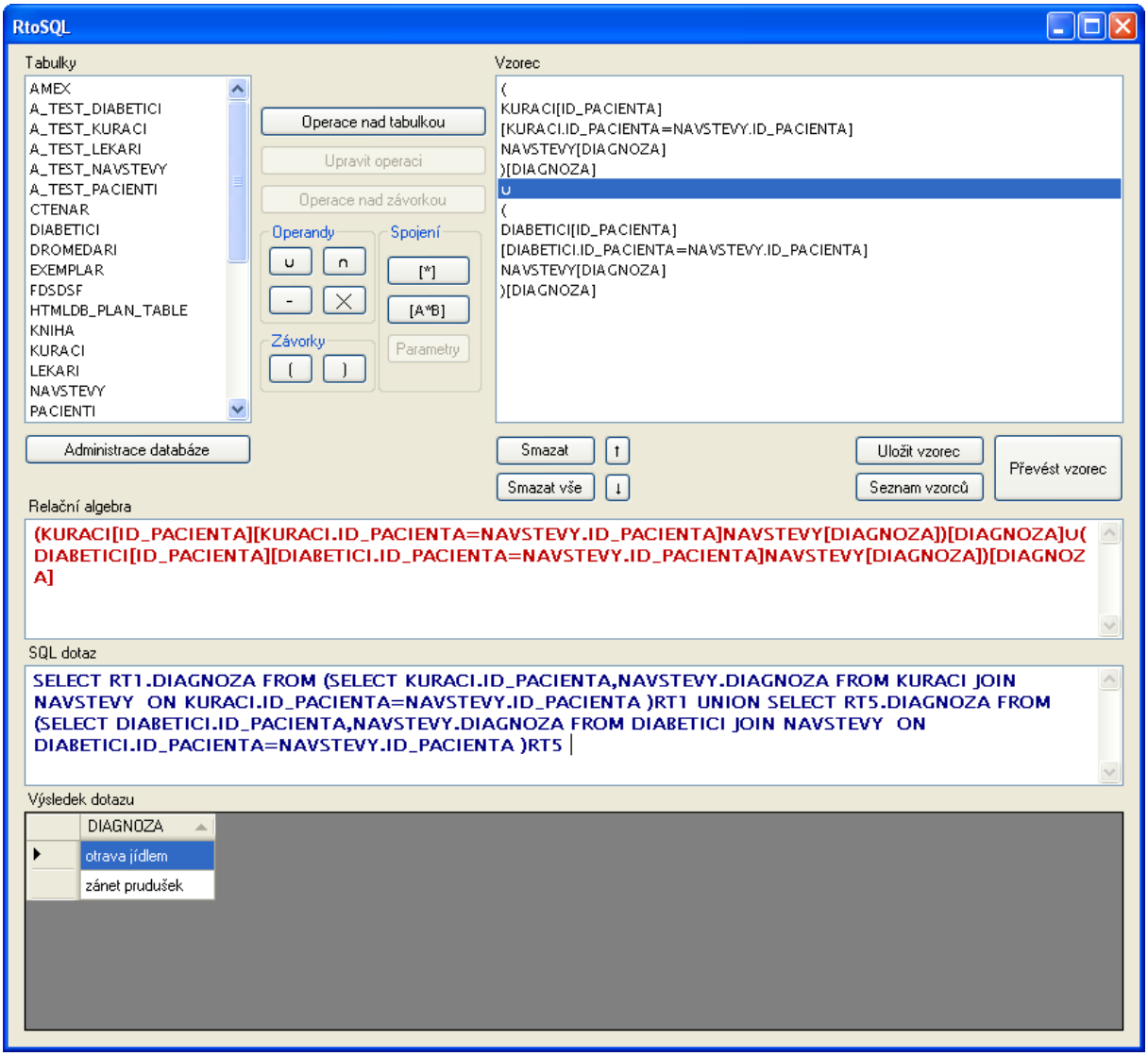
Syntaxi relační algebry jsem pro účely této aplikace mírně upravil. Pro větší přehlednost výrazů jsem se rozhodl nahradit operandy selekce a projekce závorkami za tabulkou, nad kterou je selekce a projekce prováděna. Selekce nad danou tabulkou je označena závorkami kulatými, projekce závorkami hranatými. Způsob, jakým jsou tyto dvě operace realizovány zůstává ale zcela stejný.

Pro příklady syntaxe relační algebry v programu odkazuji uživatele na kapitolu č. 10, uvádějící testované příklady relační algebry.

8.4. Uživatelské rozhraní pro vytváření výrazů relační algebry

Jedním z hlavních požadavků na výslednou aplikaci je, aby umožnila vytvářet uživateli výrazy relační algebry pomocí tvůrce výrazů. Za cíl považuji, aby byly vždy zobrazeny všechny možnosti práce s výrazem relační algebry uživatel jen dynamicky vybíral z možných cest, jak může vzorec vypadat.

Ideálním řešením tohoto je využití setříděného seznamu. Každý řádek seznamu bude obsahovat jeden operand relační algebry, jednu operaci nad tabulkou nebo závorku. Protože jednotlivé operace relační algebry jsou dostatečně známy a popsány, není vhodné, aby uživateli bylo umožněno vpisovat vlastní části výrazů relační algebry, které by mohly negativně ovlivnit samotný překlad výrazu relační algebry a ovlivnit korektnost výsledného SQL příkazu. Uživatel by tedy měl vkládat vždy celé řádky do seznamu a pouze ovlivňovat jejich výsledné pořadí při překladu. Uživatelské rozhraní jsem vzhledem k výše popsaným požadavkům tedy navrhl takto:



Obrázek č. 16 – Uživatelské rozhraní pro vytváření výrazů relační algebry

Jak je z obrázku vidět, uživatel může snadno vkládat veškeré operandy relační algebry. Může vytvářet operace nad tabulkou ze zobrazeného seznamu tabulek, upravovat je, vytvářet operace nad závorkami. Samozřejmostí je ovlivňování pořadí výrazu relační algebry pomocí šipek a mazání jednotlivých řádků seznamu. Jednotlivé skupiny tlačítek jsem se pokusil co nejpřehledněji uspořádat, aby vzniklo prostředí, ve kterém uživatel bez problémů vytvoří i nejsložitější vzorce.

8.5. Kontrola výrazů relační algebry

Před přeložením výrazu relační algebry do SQL by měla proběhnout kontrola, zda je zadaný výraz logicky správně. Půjde o kontrolu závorek, kontrolu, zda mezi operandy relační algebry existuje nějaká operace nad tabulkou atd. Až po této kontrole bude vložený výraz relační algebry přeložen do SQL a ten poté odeslán na databázi.

8.6. Překlad výrazů relační algebry

Výše jsem naznačil, jakým způsobem bude vypadat skládání výsledných výrazů relační algebry. Ten by ale neměl smysl bez realizovaného překladu do SQL. Pokusím se tedy navrhnout, jakým způsobem bude aplikace překládat uživatelem vytvořené výrazy relační algebry.

Překlad jednoduchého výrazu

Jednoduchý výraz uvažuji jako výraz bez závorek. Nejdříve je třeba seznámit čtenáře se prioritou operací, vyplývající logicky z SQL. Rozčlením operace na dvě kategorie:

1. operace sjednocení, průniku a rozdílu
2. operace množinové a kartézský součin

Mezi těmito kategoriemi operací je rozdíl v tom, že první uvedené operace jsou realizovaná každá ve vlastní SELECT klauzuli SQL. Z tohoto můžeme usoudit, že v jednoduchém příkazu relační algebry, obsahujícím n těchto operací bude SQL obsahovat n+1 vlastních SELECT klauzulí. Na příkladu:

PACIENTI[ID_PACIENTA] - NAVSTEVY[ID_PACIENTA] ∪ TESTOVANI[ID_PACIENTA]

Tento vzorec do SQL přeložíme jako:

```
SELECT ID_PACIENTA FROM PACIENTI
MINUS
SELECT ID_PACIENTA FROM NAVSTEVY
UNION
SELECT ID_PACIENTA FROM TESTOVANI
```

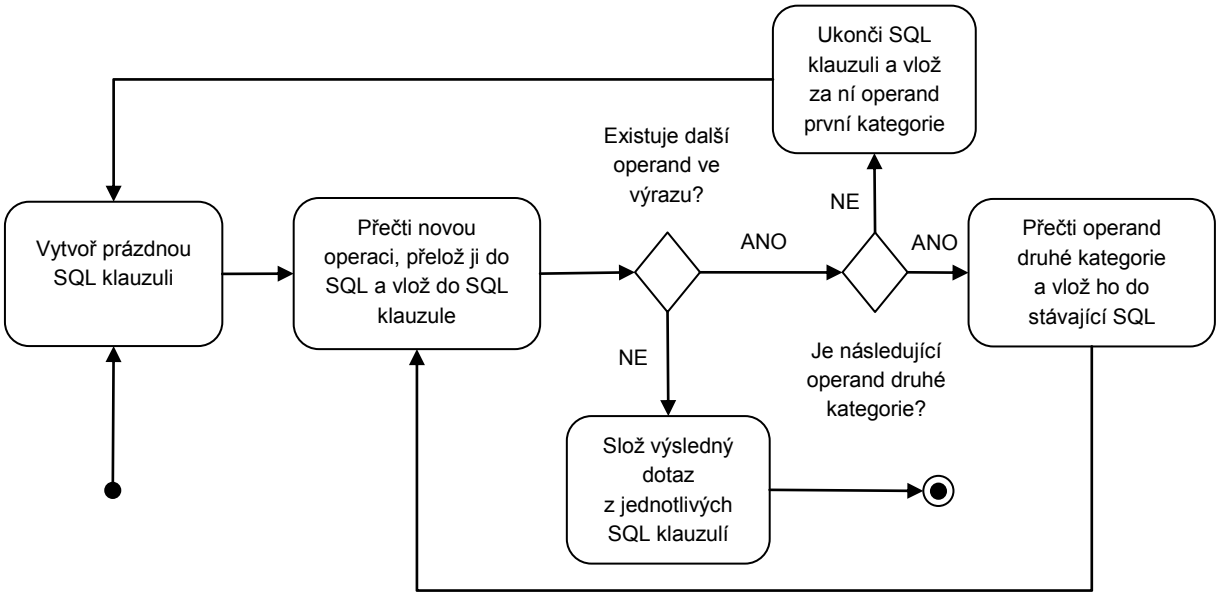
Je nutno tedy realizovat tři vlastní dotazy. Druhý typ operací se od předchozích liší. Libovolný počet pouze těchto operací je možno přeložit korektně přeložit do jednoho SQL dotazu. Na příkladu:

PACIENTI[PRIJMENI]
[PACIENTI.ID_PACIENTA=NAVSTEVY.ID_PACIENTA]
NAVSTEVY[DATUM]
[NAVSTEVY.ID_LEKARE=LEKARI.ID_LEKARE]
LEKARI[SPECIALIZACE]

Překlad tohoto výrazu do relační algebry vidíme zde:

```
SELECT PACIENTI.PRIJMENI, NAVSTEVY.DATUM, LEKARI.SPECIALIZACE
FROM PACIENTI JOIN NAVSTEVY
ON PACIENTI.ID_PACIENTA=NAVSTEVY.ID_PACIENTA
JOIN LEKARI ON NAVSTEVY.ID_LEKARE=LEKARI.ID_LEKARE
```

Výsledný překlad obsahuje jen jednu SQL klauzuli. V programu tedy budu uvažovat, že operace z druhé kategorie budou mít prioritu před operacemi z kategorie první. Pokusím se toto popsat pomocí aktivity diagramu:



Obrázek č. 17 – Popis překladu jednoduchých výrazů pomocí diagramu aktivit

Překlad složeného výrazu

Jako složený výraz uvažuji výraz obsahující závorky. Závorku, která se bude provádět jako první označím za závorku na nejvyšší úrovni. Závorka provádějící se jako poslední bude označena za závorku úrovně nejnižší.

Nejen tedy, že na něj bude aplikována logika jednoduchého překladu uvedeného výše, je nutno také tyto jednoduché překlady nějakým způsobem rozčlenit tak, aby výraz relační algebry, který bude obklopen největším počtem závorek byl řešen jako první.

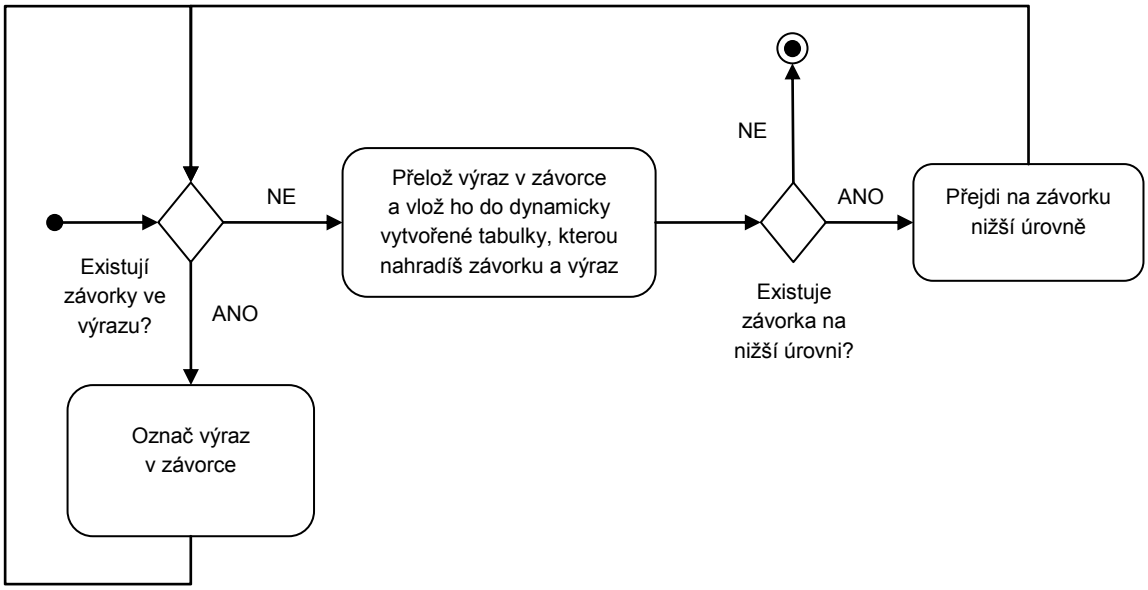
Tento výraz v závorce nahradím v SQL vlastním dotazem, který bude označen za dynamicky vytvořenou tabulku pouze pro účely daného dotazu. Uvedu na příkladu:

(PACIENTI[ID_PACIENTA] - NAVSTEVY[ID_PACIENTA])
[RT0.ID_PACIENTA=PACIENTI.ID_PACIENTA] PACIENTI

Tento výraz relační algebry obsahuje jednu závorku. Přeložený dotaz v SQL by vypadal takto:

```
SELECT * FROM
(SELECT PACIENTI.ID_PACIENTA FROM PACIENTI
MINUS
SELECT NAVSTEVY.ID_PACIENTA FROM NAVSTEVY
)RT0 JOIN PACIENTI
ON RT0.ID_PACIENTA=PACIENTI.ID_PACIENTA
```

Princip překladu se pokusím popsat následujícím diagramem aktivit:



Obrázek č. 18 – Popis překladu složených výrazů pomocí diagramu aktivit

Návrh implementace překladu

Z výše uvedeného můžu usoudit, že překlad implementuji ve dvou třídách. První třída zkontroluje, zda obsahuje výraz relační algebry závorky. Obsahuje-li, zavolá rekurzivně sám sebe. Toto se bude opakovat až do chvíle, než dojdeme k závorce na nejvyšší úrovni. Překlad této závorky bude vložen do dynamicky vytvořené tabulky a touto tabulkou bude nahrazen výraz v závorce. Opakováním dojdeme ke stavu, kdy každá závorka bude reprezentována dynamicky vytvořenou tabulkou. Nad tímto výrazem potom můžeme zavolat jednoduchý překlad. Druhá třída bude realizovat jednoduchý překlad bez závorek.

8.7. Automatické vytvoření náležitostí pro ukládání výrazů

Z důvodu co nejsnadnějšího používání výsledné aplikace by měl mít administrátor možnost jednoduše vytvořit veškeré náležitosti pro ukládání výrazů do databáze. Tento požadavek realizuji tak, že do formuláře týkající se administrace databáze vložím tlačítko pro vytvoření všech tabulek a případných sekvencí, které tabulky budou používat. Administrátor tedy pouze tímto způsobem vytvoří požadované tabulky a program již může ukládat jednotlivé vzorce a zpětně je načítat, případně mazat.

8.8. Datové typy ve vytvářených tabulkách

Je vhodné, aby byl realizován při vytváření tabulek seznam datových typů, ze kterých bude uživatel vybírat požadovaný datový typ pro konkrétní sloupec vytvářené tabulky. Protože má jít pouze o data testovací, zvolil jsem jen následující čtyři datové typy:

- VARCHAR
- INT
- DATE
- FLOAT

Tyto čtyři datové typy by měly umožnit uživateli realizovat ukládání většiny běžně zadávaných dat do databáze.

8.9. Offline překlad

Offline překlad bude realizován pomocí stejných tříd jako překlad vytvořený v uživatelském rozhraní. Rozdílem bude, že uživatel výraz relační algebry vkládá jako textový řetězec a výraz bude poté dynamicky přeložen do SQL. Nebude tedy žádným způsobem pracovat s databází. Známým problémem řešení přes textové řetězce je, že jsou velice náročná pro uživatele z pohledu dodržení syntaxe. Požadovanou syntaxi sepíšu v uživatelské příručce. Součástí offline překladače bude také několik příkladů výrazů, které si uživatel bude moci vyzkoušet vložit do překladače a přeložit.

8.10. Ukládání přihlašovacích údajů uživatele

Ve výsledné aplikaci by bylo vhodné, aby v přihlašovacím okně do databáze byla možnost uchování údajů pro přihlašování. Pokud bude aplikaci spouštět jen jeden uživatel, nebude muset znovu vyplňovat veškeré údaje. Údaj, který nepovažuji za vhodný k ukládání je heslo k databázi. Ostatní náležitosti pro připojení – typ databáze, adresa databáze, port, login, service name ukládány budou. Protože tedy neukládám žádné citlivé údaje, stačí toto uložení realizovat přes XML. Požadavkem je, aby byl XML soubor automaticky vytvářen při požadavku na uložení do stejné složky, ve které je uložena aplikace.

8.11. Vývoj aplikace

Aplikaci budu vytvářet na následujícím hardware:

- Procesor Intel Pentium 1,6 GHz
- RAM 1 GB
- Pevný disk 60 GB

Použitý software:

- Windows XP
- Microsoft Visual Studio 2008
- Oracle Database Express Edition 10.2.0.1.0
- Oracle Data Access Components for Oracle Client 11.2.0.2.1

Oracle Database nebyla nainstalována na počítači, na kterém byl vyvíjen samotný program. Databáze byla spuštěna na serveru, ke kterému se vyvíjená aplikace připojovala.

8.12. Použité technologie

Aplikaci jsem programoval pomocí Visual Studia v jazyce C#. Jako databázi jsem použil Oracle database Express.

Jazyk C#

C# je vysokoúrovňový objektově orientovaný programovací jazyk vyvinutý firmou Microsoft společně s platformou .NET. Je nepřímým potomkem jazyka C, ze kterého čerpá syntaxi. Více o jazyce C# je možno nalézt v [7].

Oracle database

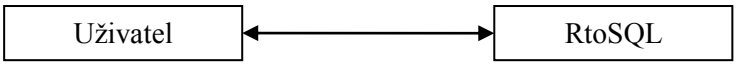
Je multiplatformním databázovým systémem. Oracle database je vyvíjena firmou Oracle Corporation. Více o Oracle database v [6].

8.13. Zavedení aplikace

Pro zavedení aplikace stačí je nutné, aby na počítači byl nainstalovaný Oracle Data Access Components for Oracle Client, pomocí něj je realizováno připojení k Oracle database. V příloze je uveden odkaz na stažení. Pokud uživatel chce pracovat nad konkrétní Oracle databází, je třeba, aby k ní měl přístup. Administrátor tedy musí definovat uživatele, nastavit jim práva a určit tabulky, ke kterým má mít ižovateů přístup. Aplikace se neinstaluje, stačí spustit výsledný exe soubor.

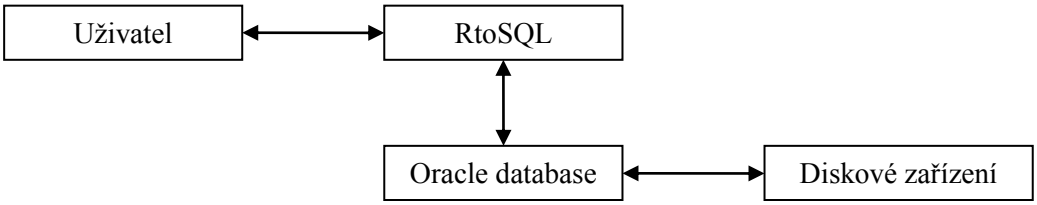
8.14. Komunikace uživatele a aplikace

Komunikaci můžeme uvažovat ve dvou vrstvách. Nepřipojí-li se uživatel ke konkrétní databázi, vypadá komunikace takto:



Obrázek č. 19 – Popis komunikace aplikace a nepřihlášeného uživatele

Je-li uživatel připojen ke konkrétní databázi, můžeme si komunikace představit takto:



Obrázek č. 20 – Popis komunikace aplikace a přihlášeného uživatele

9. Popis implementace

V této části bych měl popsat popis implementaci aplikace. Pokusím se toto co nejvíce zestručnit, protože v zadání práce je vytvoření programátorské příručky, ve které bude implementace popsána mnohem podrobněji.

9.1. Spuštění aplikace

Pro spuštění aplikace uživatel bude potřebovat následující software:

- OS Windows XP - 7
- Oracle Database Express Edition 10.2.0.1.0
- Oracle Data Access Components for Oracle Client 11.2.0.2.1

Aplikace by měla být spustitelná i s jinými srovnatelnými verzemi software. Minimální požadavky na hardware jsou tyto:

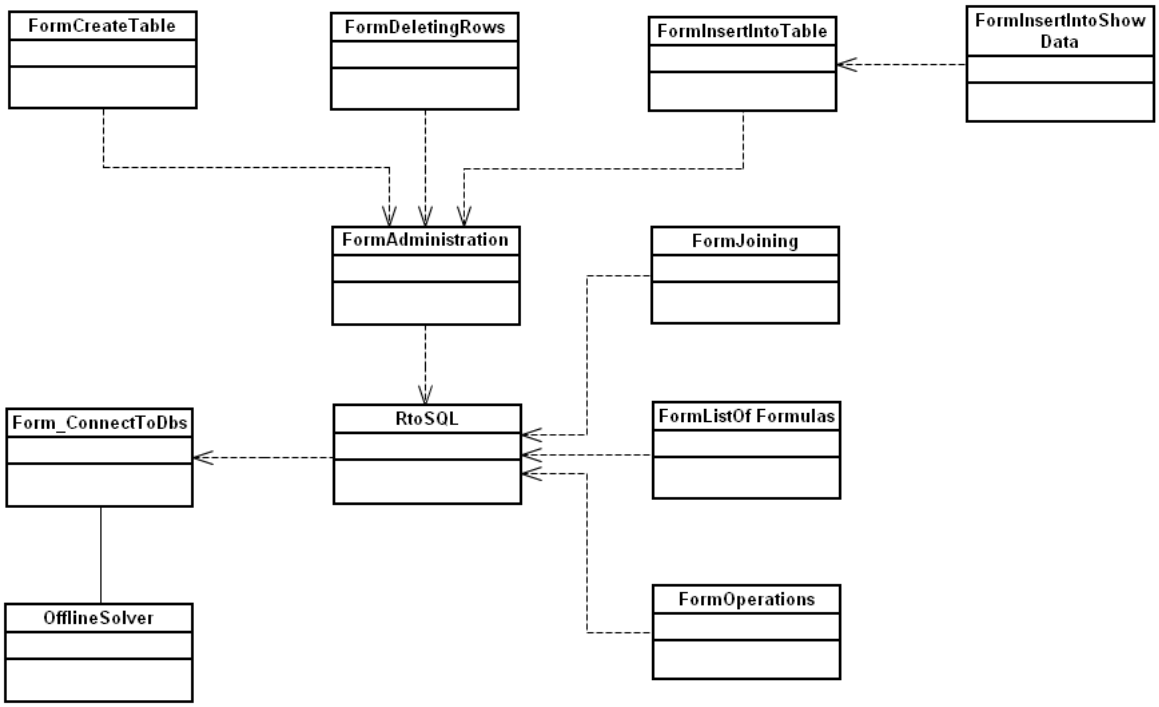
- Procesor min. 1,6 GHz
- RAM min. 1 GB
- Min. 3 MB HDD

9.2. Vzhled aplikace

Náhledy aplikace je možno nalézt v uživatelské příručce.

9.3. Popis tříd realizujících uživatelské rozhraní

Pomocí diagramu tříd se pokusím popsat komunikaci mezi jednotlivými okny aplikace. Veškerá okna, kromě formuláře realizujícího Offline překladač jsou závislá na připojení do databáze. Diagram tříd vidíme zde:



Obrázek č. 21 – Třídní diagram popisující uživatelské rozhraní

Nepopisoval jsem jednotlivé atributy a procedury tříd, třídy jsou obsáhlé a výsledek by nebyl snadno čitelný. V následujících podkapitolách popíšu, který formulář umožňuje uživateli provádět požadované úkony.

Form_ConnectToDbs

Tento formulář umožňuje uživateli přihlásit se k databázi nebo se přepnout do Offline překladače. Po vyplnění údajů je zobrazen formulář samotné aplikace RtoSQL.

OfflineSolver

Formulář obsahující vzhled a obsah Offline překladače.

RtoSQL

Hlavní formulář aplikace. Umožňuje uživateli skládat výrazy relační algebry. Odkazuje na vyplňování spojení tabulek, načítaní výrazů relační algebry, vytváření operací nad tabulkami a přepnutí se do administrátorské části programu.

FormJoining

Do tohoto formuláře se uživatel dostane při vyplňování spojení nad dvěma tabulkami.

FormListOfFormulas

Tento formulář zobrazuje seznam uložených výrazů relační algebry. Umožňuje je uživateli načítat do hlavního okna aplikace.

FormOperations

Formulář umožňující vytváření operací selekce a projekce nad zadanou tabulkou a vložení této výsledné operace do hlavního okna aplikace.

FormAdministration

V tomto formuláři získá administrátor přehled o všech dostupných tabulkách. Může dostupné tabulky mazat, vytvořit tabulky a sekvence nutné pro realizaci ukládání výrazů relační algebry, případně tyto tabulky vyprázdnit. Odkazuje na možnost vytváření tabulek, mazání záznamů a vkládání dat.

FormCreateTable

Formulář umožňující dynamicky tabulku dle zadaných parametrů.

FormDeletingRows

Umožňuje uživateli smazat vybraný záznam.

FormInsertIntoTable

Umožňuje uživateli vkládat data do vybrané tabulky. Automaticky je také zobrazeno okno se stávajícím obsahem tabulky.

FormInsertIntoTableShowData

Zobrazuje stávající obsah tabulky.

9.4. Popis tříd používaných pro překlad relační algebry

Pro překlad relační algebry do SQL jsem realizoval několik tříd. Jejich popis je uveden v programátorské příručce.

9.5. Výpis všech vytvořených souborů při implementaci

Kompletní výpis všech vytvořených souborů uvádím v programátorské příručce.

9.6. Sekvence

V Oracle database se pro generování jedinečných klíčů používá tzv. sekvencí. V databázi bude nutno vytvořit sekvence nad atributy id pro tabulku Formula a Pieces. Pomocí těchto sekvencí vygenerujeme při ukládání výrazů relační algebry do databáze vždy unikátní primární klíč.

9.7. Implementace rolí

V návrhu implementace jsem uvažoval nad rozčlenění rolí na uživatele a administrátora. Tato komponenta aplikace měla rozlišovat, zda je přihlášený uživatel dle práv administrátorem nebo uživatelem a podle toho zpřístupnit tlačítka pro administraci databáze. Tato část není nakonec ve výsledné aplikaci obsažena z důvodu velkých možností nastavování jednotlivých práv uživatele na Oracle database. Špatné vyhodnocení práv uživatele by mohlo vést k nefunkčnosti administrace databáze v programu i pro uživatele, kteří práva na úpravu databáze mají.

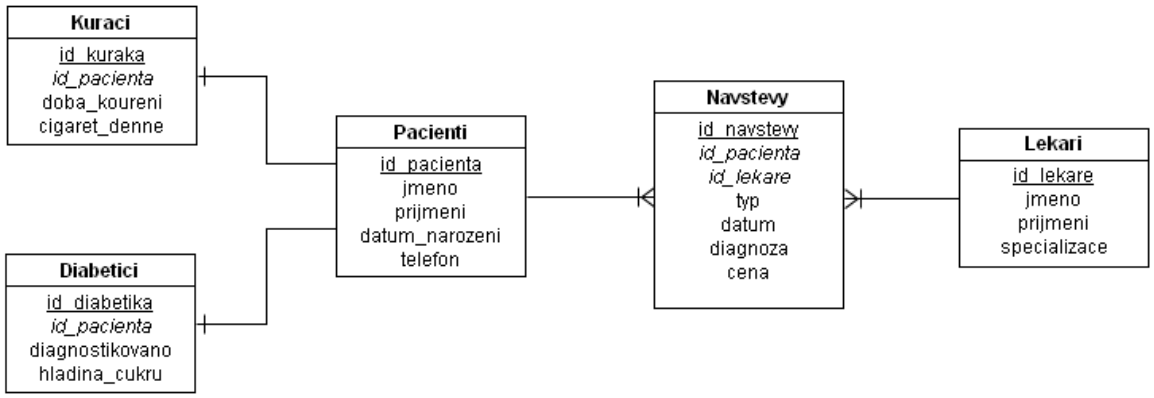
Je tedy pouze na administrátorovi databáze, jakým způsobem zpřístupní na úrovni SŘBD konkrétním uživatelům data. Pokud se přihlásí uživatel, který například nemá práva na ukládání výrazů, aplikace bude zobrazovat chybová hlášení při pokusech jeho o ukládání.

10. Testování

V této kapitole se zaměřím na testování vytvořeného programu. Půjde především o možnost práce s databází a samotný překlad výrazů, kdy se pomocí programu pokusím vytvořit několik propojených tabulek, naplnit je daty a volat nad nimi dotazy v relační algebře. Budu sledovat, jak korektní budou výsledné vzorce a zda zadání dotazu bude odpovídat výsledné relaci.

10.1. Testovací databáze

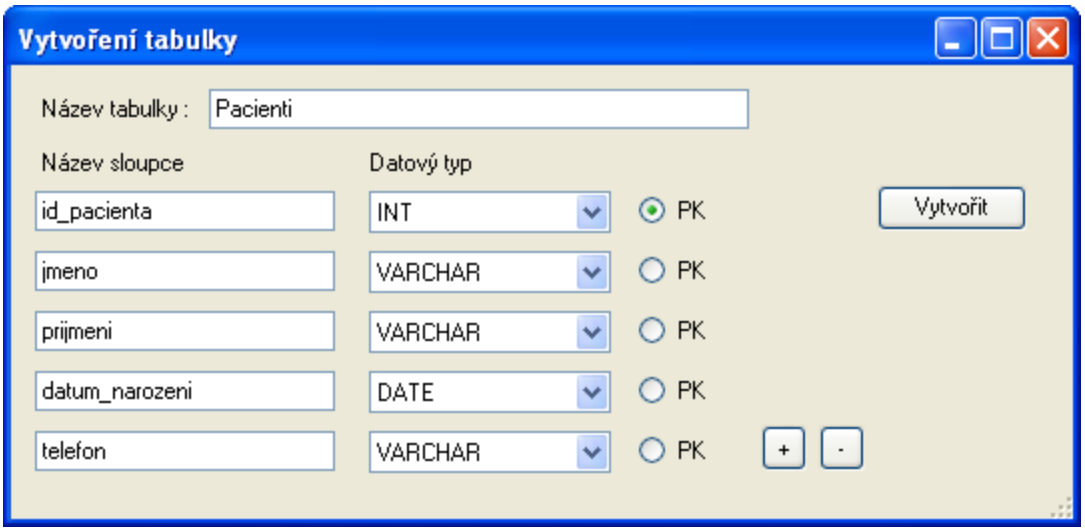
Pro test jsem vymyslel databázi, která bude zaznamenávat seznam pacientů, seznam lékařů, seznam návštěv pacientů u lékařů a seznam pacientů s cukrovkou a kuřáků. Cílem databáze je sledovat případné častější návštěvy diabetiků a kuřáků u specializovaných lékařů. Pomocí ERD zobrazíme databázi takto:



Obrázek č. 22 – Databáze pro testování překladu výrazů

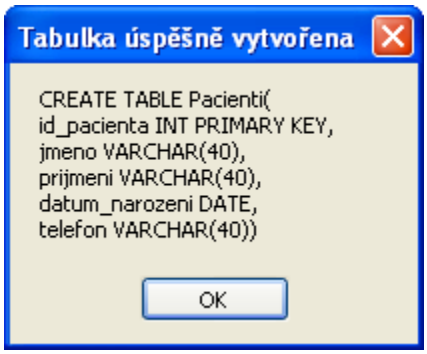
10.2. Vytváření tabulek

Vytvoření tabulek dle zadání proběhlo zcela bez problémů. Na obrázku je vidět, jak v aplikaci vypadá vytváření tabulek:



Obrázek č. 23 – Formulář pro vytvoření tabulky

Informace o úspěšném vytvoření tabulky byla zobrazena tímto způsobem:

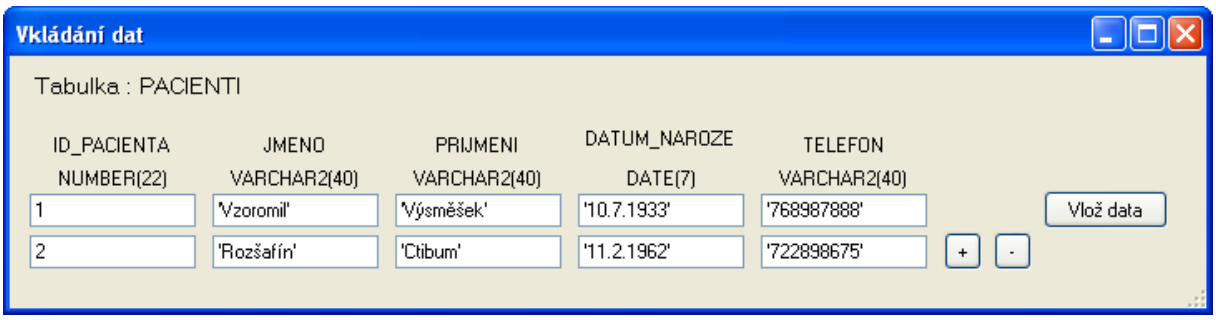


Obrázek č. 24 – Informace o vytvoření tabulky

Tímto způsobem jsem vytvořil všechny tabulky dle zobrazeného ER diagramu. Vyzkoušel jsem všechny dostupné datové typy. Uživatel veden uživatelskou příručkou by neměl mít s vytvářením tabulek žádný problém. Tato část testu tedy dopadla úspěšně.

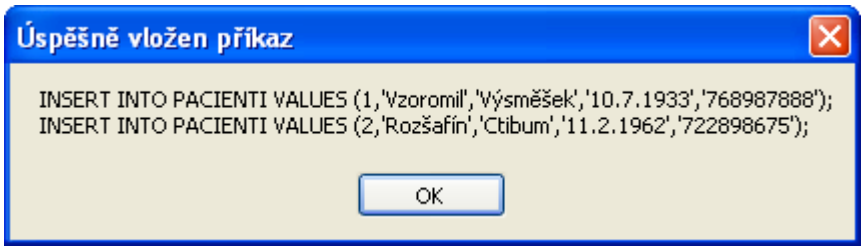
10.3. Vkládání dat

Tabulky jsem úspěšně naplnil daty, nevyskytla se žádná chyba. Nezkoušel jsem vkládat velká množství dat, očekává se, že při testování na velkých databázích budou data do databáze vkládána např. přes funkci BULK LOAD. Zde je ukázka vložení dvou záznamů do tabulky Pacienti:



Obrázek č. 25 – Formulář pro vkládání záznamů

A po kliknutí na tlačítko pro vložení program zobrazil úspěšné vložení informačním oknem, které vypadalo takto:



Obrázek č. 26 – Informace o vložení záznamů

V této části testu se tedy také neobjevil žádný problém, test dopadl úspěšně.

10.4. Testování překladu

Tato část se zabývá samotným překladem výrazů z relační algebry do SQL. Otestuji několik triviálních i netriviálních dotazů. Jednotlivé části výsledků testu se budou skládat z textové formy dotazu, kterou by uživatel mohl vznést k této databázi. Dále vzorec relační algebry, který reprezentuje tento dotaz. Poslední částí je výsledek překladu relačního vzorce do SQL.

Testovací dotaz č. 1

Textový popis	Uživatel požaduje zobrazení obsahu tabulky Pacienti.
Vzorec v RA	*PACIENTI*
Překlad do SQL	`SELECT * FROM PACIENTI`

Testovací dotaz č. 2 – selekce, projekce

Textový popis	Uživatel požaduje zobrazení sloupců jména a data narození ze seznamu pacientů všech pacientů, kteří se narodili později než v roce 1990.
Vzorec v RA	*PACIENTI(DATUM_NAROZENI > '1.1.1990')[PRIJMENI, DATUM_NAROZENI]*

Překlad do SQL	`SELECT PACIENTI.PRIJMENI, PACIENTI.DATUM_NAROZENI`
	`FROM PACIENTI`
	`WHERE PACIENTI.DATUM_NAROZENI > '1.1.1990'`

Testovací dotaz č. 3 – přirozené spojení

Textový popis	Uživatel požaduje zobrazení sloupců telefonu pacienta a ceny jeho ošetření u pacientů, kteří byli někdy ošetřeni ambulantně.
Vzorec v RA	*PACIENTI[TELEFON][*]NAVSTEVY(TYP > 'ambulantně')[CENA]*

Překlad do SQL	`SELECT TELEFON, CENA`
	`FROM PACIENTI`
	`NATURAL JOIN`
	`NAVSTEVY`
	`WHERE TYP > 'ambulantně'`

Testovací dotaz č. 4 – obecné spojení

Textový popis	Uživatel požaduje zobrazení jména pacienta, data jeho návštěvy a specializaci lékaře, kterého navštívil.
Vzorec v RA	*PACIENTI[PRIJMENI][PACIENTI.ID_PACIENTA=NAVSTEVY.ID_PACIENTA]* *NAVSTEVY[DATUM][NAVSTEVY.ID_LEKARE=LEKARI.ID_LEKARE]* *LEKARI[SPECIALIZACE]*

Překlad do SQL	`SELECT PACIENTI.PRIJMENI, NAVSTEVY.DATUM,`
	`LEKARI.SPECIALIZACE`
	`FROM PACIENTI JOIN NAVSTEVY`
	`ON PACIENTI.ID_PACIENTA=NAVSTEVY.ID_PACIENTA`
	`JOIN LEKARI`
	`ON NAVSTEVY.ID_LEKARE=LEKARI.ID_LEKARE`

Testovací dotaz č. 5 – odečítání

Textový popis	Uživatel požaduje zobrazení všech pacientů, kteří neabsolvovali žádnou návštěvu a informací o nich.
Vzorec v RA	*(PACIENTI[ID_PACIENTA]-NAVSTEVY[ID_PACIENTA])* *[RT0.ID_PACIENTA=PACIENTI.ID_PACIENTA]PACIENTI*

Překlad do SQL	`SELECT * FROM`
	`(SELECT PACIENTI.ID_PACIENTA FROM PACIENTI`
	`MINUS`
	`SELECT NAVSTEVY.ID_PACIENTA FROM NAVSTEVY`
	`)RT0`
	`JOIN PACIENTI`
	`ON RT0.ID_PACIENTA=PACIENTI.ID_PACIENTA`

Testovací dotaz č. 6 – průnik

Textový popis	Uživatel požaduje získání seznamu návštěv všech pacientů, kteří mají vysokou hladinu cukru a jsou i silní kuřáci a specializaci lékaře, který je ošetřoval.
Vzorec v RA	*(KURACI(CIGARET_DENNE>20)[ID_PACIENTA]∩ DIABETICI(HLADINA_CUKRU>4.8)[ID_PACIENTA]) [RT0.ID_PACIENTA=NAVSTEVY.ID_PACIENTA]NAVSTEVY[ID_LEKARE] [NAVSTEVY.ID_LEKARE=LEKARI.ID_LEKARE]LEKARI[SPECIALIZACE]*
Překlad do SQL	`SELECT RT0.*,NAVSTEVY.ID_LEKARE,LEKARI.SPECIALIZACE FROM (SELECT KURACI.ID_PACIENTA FROM KURACI WHERE KURACI.CIGARET_DENNE > 20 INTERSECT SELECT DIABETICI.ID_PACIENTA FROM DIABETICI WHERE DIABETICI.HLADINA_CUKRU > 4.8)RT0 JOIN NAVSTEVY ON RT0.ID_PACIENTA=NAVSTEVY.ID_PACIENTA JOIN LEKARI ON NAVSTEVY.ID_LEKARE=LEKARI.ID_LEKARE`

Testovací dotaz č. 7 – sjednocení, závorkování

Textový popis	Uživatel požaduje získání seznamu diagnóz všech kuřáků a diabetiků
Vzorec v RA	*(KURACI[ID_PACIENTA] [KURACI.ID_PACIENTA=NAVSTEVY.ID_PACIENTA] NAVSTEVY[DIAGNOZA]) [DIAGNOZA] ∪ (DIABETICI [ID_PACIENTA] [DIABETICI.ID_PACIENTA=NAVSTEVY.ID_PACIENTA]NAVSTEVY [DIAGNOZA]) [DIAGNOZA]*
Překlad do SQL	`SELECT RT0.DIAGNOZA FROM (SELECT KURACI.ID_PACIENTA,NAVSTEVY.DIAGNOZA FROM KURACI JOIN NAVSTEVY ON KURACI.ID_PACIENTA=NAVSTEVY.ID_PACIENTA)RT0 UNION SELECT RT3.DIAGNOZA FROM (SELECT DIABETICI.ID_PACIENTA,NAVSTEVY.DIAGNOZA FROM DIABETICI JOIN NAVSTEVY ON DIABETICI.ID_PACIENTA=NAVSTEVY.ID_PACIENTA)RT3`

10.5. Závěr testování

Výše zmíněné výrazy relační algebry jsem postupně nechal přeložit výslednou aplikací. Snažil jsem se zvolit vzorce, které budou reprezentovat obvyklé typy dotazů. Zvolené vzorce relační algebry aplikace korektně přeložila, celé testování tedy proběhlo bez chyb. Výsledky překladu do SQL jsou přímo zkopírovány z aplikace. Výsledný kód je jen naformátován pro snadnější čtení.

11. Porovnání s existujícími aplikacemi

Jedním z bodů zadání je popis porovnání existujících aplikací s mnou vytvořeným programem. Programů pro překlad relační algebry do SQL je celá řada, větší část z nich ale nebylo možno otestovat kvůli složitosti jejich zprovoznění. Vybral jsem následující:

- Relational Algebra Translator
- Relational Algebra Learning Tool

11.1. Relational Algebra Translator

RAT (Relational Algebra Translator) je nástroj pro překlad relační algebry do SQL vyvinutý na Národní univerzitě v Kostarice. V současné době existuje ve verzi 4, která je i s dokumentací dostupná na oficiálních stránkách univerzity.

RAT umožňuje připojení k velkému množství různých databází, např. Oracle, MySQL, Postgres. Definuje všechny operace relační algebry kromě jednotlivých polospojení. Výraz relační algebry je poté také vykreslen v binárním stromu, který ukazuje, v jakém pořadí se budou jednotlivé operace provádět.

Psaní výrazů je ale v programu dosti složité, probíhá v textové formě a překlad nebyl vždy korektní, u mého testování se například vyskytla chyba u sjednocení a průniku na stejné úrovni. Také se mi nepodařilo zjistit, zda je možné výrazy relační algebry závorkovat.

Na rozdíl od mého zadání, tedy vytváření výrazů relační algebry pomocí tvůrce výrazů je nutno do RAT výrazy ručně psát. Zároveň komunikuje s databází jen na úrovni zasílání výsledných SQL příkazů, není tedy možno zjistit, jaké sloupce například obsahuje konkrétní tabulka.

11.2. Relational Algebra Learing Tool

RALT (Relational Algebra Learning Tool) je nástroj pro vytvoření dotazů v relační algebře v grafickém prostředí. Je pevně propojen s databází, ke které se při spuštění programu připojuje. Umožňuje připojení k databázím MSSQL Server 2005 a Postgres 7.4. Aplikace je vytvořena v Javě.

Uživatel vkládá jednotlivé operace do grafického prostředí a přiřazuje jim tabulky, nad kterými volí požadovanou projekci a selekci. Uživatel tedy vytváří diagram jednotlivých operací, který je poté přeložen. U tabulek jsou zároveň zobrazeny i jednotlivé sloupce, jejich datové typy a primární klíče. Je zobrazována i výsledná relace dle zadaného výrazu relační algebry.

Překlad vypadá velice spolehlivě, testoval jsem aplikaci s MSSQL Server a nevyskytl se žádný problém. Aplikace je velice podobná aplikaci RtoSQL, kterou jsem implementoval, rozdílný je přístup k návrhu výrazů, kdy mnou vytvořená aplikace nepracuje s podobným grafickým vkládáním výrazů. RALT neimplementuje offline překlad jako mnou implementovaná aplikace RtoSQL. Více o RALT v [8].

12. Závěr

Cílem této bakalářské práce bylo navrhnutí a implementace nástroje pro překlad výrazů relační algebry do SQL.

V teoretické části jsem probral jednotlivé příkazy relační algebry a popsal překlad operací do SQL nad zvoleným SŘBD. Dále jsem podrobně rozebral požadavky na program a jeho funkce. Tento program jsem implementoval ve zvoleném programovacím jazyce a otestoval překlad nad jednoduchými i složitějšími dotazy. Při testování se aplikace velice osvědčila, nedošlo k žádnému nestandardnímu chování a překlad byl vždy korektní a funkční. K implementovanému programu jsem také sepsal uživatelskou a programátorskou příručku. Výsledný program jsem dále porovnal s jinými podobnými aplikacemi, při srovnávání jejich funkcí vyšlo najevo, že přestože některé jejich funkce mnou implementovaný program neobsahoval, v jiných je předčil.

Všechny body bakalářské práce se mi tímto podařilo splnit, výsledná aplikace je plně funkční. Do budoucna by mohla být rozšířena o spolupráci s větším množstvím SŘBD, případně přetvořena do webové aplikace a umístěna na školní server, kde by mohla být používána k lepšímu pochopení principů a syntaxe relační algebry v rámci výuky databázových systémů.

13. Literatura

[1] POKORNÝ, Jaroslav. *Dotazovací jazyky*. 3. vyd. Veletiny: Science, 1994, 225 s. ISBN 80-901-4752-6.

[2] ŠARMANOVÁ, Jana. *Teorie zpracování dat* [online]. Vyd. 2., přeprac. Ostrava: Vysoká škola báňská - Technická univerzita, 2007 [cit. 2012-05-01]. ISBN 978-80-248-1498-8. Dostupné z: http://www.elearn.vsb.cz/archivcd/FEI/TZD/TZD.pdf

[3] DIETRICH, Suzanne Wagner. *Understanding relational database query languages*. Upper Saddle River, NJ: Prentice Hall, 2001, 176 s. ISBN 01-302-8652-4.

[4] ŠARMANOVÁ, Jana. *Databázové a informační systémy* [online]. 2007 [cit. 2012-05-01]. Studijní text. Dostupné z: http://www.elearn.vsb.cz/archivcd/FEI/DAIS/DAIS.pdf

[5] Diagram datových toků. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-05-01]. Dostupné z: http://cs.wikipedia.org/wiki/Diagram_datových_toků

[6] DAWES, Chip a Biju THOMAS. *OCA/OCP*: *introduction to Oracle9i SQL : study guide* [online]. 1st ed. San Francisco, CA: Sybex, 2002, 539 s. [cit. 2012-05-01]. ISBN 07-821-4062-9.

[7] TROELSEN, Andrew W. *Pro C# 2008 and the .NET 3.5 platform*. 4th ed. New York: Distributed to the book trade worldwide by Springer-Verlag New York, 2007, 1370 s. ISBN 15-905-9884-9.

[8] MITRA, Pritam. *Relational Algebra Learning Tool* [online]. London, 2009 [cit. 2012-05-01]. Dostupné z: http://www.doc.ic.ac.uk/~pjm/teaching/student_projects/pm105_report.pdf. Imperial College London.

14. Seznam příloh

Na CD se jako příloha nachází tyto soubory:

Zadání bakalářské práce
Bakalářská práce
Aplikace RtoSQL
RtoSQL solution
Kompletní DFD
Programátorská příručka
Uživatelská příručka